Universidad de las Ciencias Informáticas
Facultad 6

Título: BioSyS: Implementación del Módulo de Análisis.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor(es): Yiliana Rodríguez Castillo

Yuandry Noa Gálvez

Tutor(es): Msc. Noel Moreno Lemus

Ciudad de La Habana, Junio del 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yiliana Rodríguez Castillo

_____

Firma del Autor

Yuandry Noa Gálvez

_____

Firma del Autor

Noel Moreno Lemus

_____

Firma del Tutor

AGRADECIMIENTOS

A nuestros familiares por habernos apoyado durante toda la carrera.

A nuestro tutor Noel por su comprensión y paciencia.

A Carrasco por sus consejos y sabiduría.

A Liudmila por toda su ayuda incondicional.

A Roberto y Adisbel por estar siempre a nuestro lado en las buenas y en las malas.

A Edel, Luis, Hermes, Haymee, Martica, Manolo, Carrazana, Tita, Irylis, Mabel, Yanelis, Rene, Adriel, Camilo y su amigo.

A todos nuestros amigos que nos han escuchado, aconsejado y brindado su apoyo en los momentos difíciles.

A los que de manera involuntaria dejamos de mencionar, nuestro más sincero agradecimiento.

DEDICATORIA

De Yiliana Rodríguez Castillo

A mis padres: Ana María y Alfredo, especialmente a mi madre por tener confianza en mí, y por haberme dado fuerzas para terminar lo que hace 5 años comencé y hoy con mucho sacrificio termino.

A mi tío Francisco Vinent y a Cuellito que han sido como un padre para mí.

A mis abuelita Gladys y Lula, y a mi tía Yakelín, las quiero mucho.

A dos mujeres super especiales: mis tías Tania Elsi y María Magalys, quiero que sepan que significan mucho para mí.

A mis hermanos y primos, especialmente a Liette.

A toda mi familia y seres queridos.

A mis amigos de la Universidad, en especial a Yeli, que ha sido mi paño de lágrimas y ha tenido que soportarme todo este tiempo.

A todos mis amigos de Santiago, Yanne, Isa, Leivi, Rachel, Yile, Luisa, Emi y Jorge.

A mi novio Robe, por su comprensión, paciencia, apoyo y cariño en todo momento.

A mi compañero y hermano Yuandry que me ha demostrado que existe la verdadera amistad.

De Yuandry Noa Gálvez

A mis padres: Magalis y Lázaro a los que debo lo que soy, que me han apoyado todos estos años, han tenido confianza en mí y sobre todo me han brindado su amor.

A Marlenis, Luis y Nancy que han sido como padres para mi, que me han ayudado en momentos difíciles y he podido contar con ellos todos estos años de sacrificio.

A mis amigos y hermanos Mairelis, Vicen y Alberto que han sido un ejemplo a seguir todos estos años.

A mis familiares que siempre me han querido y apoyado en especial mis tías, tíos, abuelos y especialmente a mis primos.

A mis amigos de la Universidad: Que me han apoyado, comprendido y aceptado durante estos 5 años y a los que nunca olvidaré.

A mis amigos de Ciego, que a pesar de la distancia siempre puede contar con ellos.

A mi amigo Aníbal: Al que le debo lo que soy, mis ambiciones de estudiante, mi carrera, mi sacrificio, gracias a su ejemplo y al de su familia.

A Nuemy, que este título también es de ella, a mi tía Maricel a las dos Mei Ling y a Alfre.

A mi amiga y novia: Adis por su apoyo a la cual le debo estar aquí hoy graduado, la que me guio siempre por el mejor camino de una forma u otra, y a sus Padres y Abuelos a los que quiero y admiro.

A mi país, a mi patria y a Fidel, a los que debo lo que soy y lo que seré.

A mi compañera y hermana Yiliana gracias a la cual me he podido graduar, por su comprensión, paciencia y dedicación al trabajo y por la gran amistad que hemos logrado construir durante todos estos años.

RESUMEN

BioSyS es un software para la Simulación de Sistemas Biológicos desarrollado de conjunto por especialistas del Centro de Inmunología Molecular y la Universidad de las Ciencias Informáticas. El mismo, cuenta con un potente Módulo de Simulación que hace uso de tecnologías Grid de computación para hacer exploraciones intensivas sobre modelos matemáticos descritos mediante sistemas de ecuaciones diferenciales. Los resultados de estas simulaciones se almacenan en una Base de Datos. El objetivo del Módulo de Análisis es precisamente realizar análisis de estos resultados. En la versión 1.0 de BioSyS se implementaron funcionalidades muy básicas de análisis y en el presente trabajo se desarrolla una nueva versión del Módulo de Análisis del software BioSyS, que mejora las funcionalidades de la versión 1.0 e incluye nuevas funcionalidades que cumplan con los requisitos del cliente y brinden al investigador una herramienta poderosa para el análisis, que cuente además con una interfaz amigable.

Palabras Claves: BioSyS, Simulación, Resultados, Análisis, Funcionalidades, Herramienta, Interfaz, Algoritmo.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Los estudios en el área de la Biología han alcanzado niveles extraordinarios en los últimos años, lo que ha propiciado que se acumule mucha información de ahí la importancia que ha alcanzado la Bioinformática para almacenar y manejar el flujo cada vez mayor de datos biológicos, relacionada con los componentes de los sistemas biológicos, genes, proteínas, moléculas pequeñas y demás. Este conocimiento ha hecho que los científicos vayan modificando sus paradigmas de investigación y traten de entender el funcionamiento de los sistemas como un todo, dando paso así al nacimiento de la Biología de Sistemas.

La Biología de Sistemas es un área de investigación científica que se preocupa del estudio de procesos biológicos usando un enfoque sistémico, que tiene dos componentes fundamentales: uno experimental y uno computacional. El lado experimental proporciona los datos necesarios para la creación y validación de los modelos matemáticos. En cambio el lado computacional está dividido en dos grandes grupos, la modelación y la minería o análisis de los datos ya existentes o de los generados durante las simulaciones. [1]

Estos datos son generados por problemas biológicos que se presentan a diario. Para realizar estudios de estos sistemas, comúnmente se crean modelos matemáticos que lo representen. Estos modelos equivalen a una ecuación matemática o un conjunto de ellas en base a las cuales se puede conocer el comportamiento del sistema. Los modelos matemáticos pueden ser por ejemplo sistemas de ecuaciones diferenciales (SED). Estos son ampliamente utilizados en la modelación de problemas biológicos, debido a las facilidades de modelación y de simulación que brindan, además de que pueden describir gran cantidad de procesos biológicos.

Al tratar de resolver estos modelos matemáticos, haciendo uso de métodos numéricos se debe suministrar a dichos métodos las condiciones iniciales, el tiempo de integración y los valores de los parámetros que forman el SED. Como resultado de la integración numérica se obtienen una Serie Temporal que describe el comportamiento aproximado del sistema. Cuando se quiere comprender a profundidad el comportamiento de uno de estos sistemas se hace necesario realizar exploraciones intensivas, que consisten en definir diferentes conjuntos de parámetros, condiciones iniciales y tiempos de integración. La necesidad de interpretar estos resultados ha sido un problema para la comunidad científica.

Para resolver este problema, en las últimas décadas se han desarrollado un gran número de aplicaciones computacionales que pretenden facilitar la modelación y el análisis de sistemas biológicos. La mayor desventaja que existe en estas aplicaciones es que se han centrado mucho en la

modelación y la simulación, mientras que los análisis de los resultados son muy básicos, dejando el peso de los mismos en manos de los usuarios, quienes tienen que ingeniárselas para obtener información valiosa de los estudios realizados. Además, muchos de los software existentes no están disponibles para la comunidad científica, debido a que están representados por grandes intereses comerciales.

En Cuba, desde hace algunos años se han tomado varias medidas para fomentar el estudio y desarrollo de distintas ramas de las Ciencias Biológicas, por lo que, en la década del 80 comienzan a crearse centros biotecnológicos donde se destaca el Centro de Investigaciones Biológicas, el Centro de Ingeniería Genética y Biotecnología (CIGB), Centro de Producción de Animales de Laboratorio (CENPALAB), Centro Nacional de Bio-Preparados (BIOCEN), Centro de Inmuno-Ensayo y el Centro de Inmunología Molecular (CIM). Estos centros y sus investigadores no están exentos de los problemas que se han venido mencionando. Es por ello que investigadores del CIM y de la UCI se dieron a la tarea de desarrollar un software para el estudio de sistemas biológicos, llamada BioSyS.

En la primera versión del software BioSyS se incluyó un Módulo de Análisis que hacía uso de diferentes técnicas o métodos como por ejemplo: Dinámica de Población, Clustering, Clasificación y Definición de grupos de reglas para posteriores análisis. Estas variantes de análisis, principalmente las tres últimas resultan muy novedosas. Aunque las mismas no se han implementado de forma detallada en la primera versión de BioSyS, esta implementación se ha hecho más bien con el objetivo de demostrar su utilidad en este tipo de estudios. No obstante, para la implementación de estos métodos, no se ha hecho un minucioso levantamiento de requisitos, ni se ha hecho un buen diseño de la interfaz del sistema.

Además, de la utilización de estas técnicas han surgido nuevas funcionalidades basadas en el uso de las mismas, que deben ser incorporadas en una nueva versión. Estas nuevas funcionalidades serán explicadas en el capítulo 4.

Por lo que se puede plantear como **problema científico**: ¿Cómo implementar el Módulo de Análisis de Series Temporales generadas en los procesos de simulación de sistemas biológicos implementados en el software BioSyS?

La investigación está centrada en realizar el Análisis a las Series Temporales que se obtienen en las simulaciones, por tanto planteamos como **Objeto de Estudio**: Análisis de Series Temporales.

Debido a que esta es un área, donde convergen muchos estudios científicos, nuestro **Campo de Acción** se sitúa exactamente en el**:** Análisis de las Series Temporales generadas en los procesos de simulación de sistemas biológicos implementados en el software BioSyS.

Como consecuencia, de que los software disponibles no poseen técnicas profundas para la realización de análisis, se plantea como **Objetivo General:** Implementar el Módulo de Análisis de Series Temporales generadas en los procesos de simulación de sistemas biológicos en el software BioSyS.

De este objetivo general se derivan los siguientes **Objetivos Específicos:**

1. Identificar requisitos del Módulo de Análisis de Series Temporales.
2. Diseñar el Módulo de Análisis de Series Temporales.
3. Implementar el Módulo de Análisis de Series Temporales.

Para dar cumplimiento a los objetivos que no hemos trazado debemos cumplir con las siguientes **Tareas:**

1. Familiarización y profundización en el estado del arte del tema.
2. Selección de los algoritmos ya existentes que resuelven algunas de las funcionalidades que tendrá el Módulo.
3. Obtención, descripción y validación del Módulo.
4. Implementación de las funcionalidades del sistema.

EL PRESENTE TRABAJO DE DIPLOMA ESTÁ ESTRUCTURADO EN 4 CAPÍTULOS

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA: En este capítulo se explicarán los conceptos fundamentales que rodean el Análisis de Series Temporales (AST) para mejorar la comprensión de estas, partiendo de las simulaciones de donde se obtienen las mismas. Se ejemplificarán software que realiza AST, llegando así a la parte más importante del capítulo donde se explican los diferentes métodos para el AST en BioSyS. Finalizando con una breve explicación de las metodologías y herramientas que posibilitan que el presente trabajo produzca los resultados esperados por los clientes.

CAPÍTULO 2: CARACTERISTICAS DEL SISTEMA: En el presente capítulo se analizarán las características del sistema, aquí se abordarán aspectos relevantes como son: las condiciones o capacidades que el software debe cumplir (requisitos funcionales), así como las propiedades o cualidades que el producto debe tener (Requisitos no funcionales). Además se definirán actores que interactúan con el sistema, los casos de uso del sistema, con sus respectivas descripciones y finalmente la relación existente entre actores y casos de uso (Diagrama de Casos de Uso del Sistema).

CAPÍTULO 3: DISEÑO DEL SISTEMA: En este capítulo se agrupan los artefactos y descripciones necesarias para que el diseño de la aplicación este guiado por un proceso organizado, que involucra Patrones de Diseño y Arquitectura muy importantes que definen la estructura de nuestro software. Finalmente se presentan los Diagramas de Clases del Diseño e Interacción que muestran las clases con sus relaciones y detalles internos.

CAPÍTULO 4: IMPLEMENTACIÓN DEL SISTEMA: Este capítulo contiene las principales características de la implementación del Sistema. Además de los principales componentes y sus relaciones, mostrados de forma sencilla a través del Diagrama de Componentes. También se muestran los resultados obtenidos con el código fuente y las interfaces más importantes de las principales clases empleadas en la implementación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se explicarán los conceptos fundamentales que rodean el Análisis de Series Temporales (AST) para mejorar la comprensión de estas, partiendo de las simulaciones de donde se obtienen las mismas. Se ejemplificarán software que realizan AST, llegando así a la parte más importante del capítulo donde se explican los diferentes métodos para el Análisis en BioSyS. Finalizando con una breve explicación de las metodologías y herramientas que posibilitan que el presente trabajo produzca los resultados esperados por los clientes.

1.1 Simulación de Sistemas Biológicos

La Simulación surge desde la década del 40 como una forma de modelar situaciones de la vida real, es una técnica numérica para reproducir artificialmente un fenómeno o el funcionamiento de un sistema, utilizando modelos matemáticos y lógicos, específicos para cada proceso, que describen el comportamiento y la estructura de dicho fenómeno a través del tiempo. La simulación es una poderosa herramienta que es ampliamente utilizada por los científicos que estudian los sistemas complejos. [2] "Simular, es reproducir artificialmente un fenómeno o las relaciones entrada−salida de un sistema". [2] Según Thomas H. Naylor y R. Bustamante:

"Simulación es una técnica numérica para conducir experimentos en una computadora digital. Estos experimentos comprenden ciertos tipos de relaciones matemáticas y lógicas, las cuales son necesarias para describir el comportamiento y la estructura de sistemas complejos del mundo real a través de largos periodos de tiempo" [3]. El estudio de los **Sistemas Biológicos** se hace más factible mediante la creación de un modelo matemático que lo describa.

1.1.1 Modelos Matemáticos

En la actualidad se utilizan los modelos matemáticos en la mayoría de los campos de la ingeniería. Un **Modelo Matemático** se define como una descripción desde el punto de vista de las matemáticas de un hecho o fenómeno del mundo real. El objetivo del modelo matemático es entender ampliamente el fenómeno y tal vez predecir su comportamiento en el futuro. Como por ejemplo: puede ser el crecimiento de las poblaciones, la concentración de un producto en una reacción química, el funcionamiento de las neuronas y la dinámica intracelular, por citar solo algunos ejemplos de su aplicación en biología.

Es muy importante tener en cuenta que un modelo matemático nunca es una representación exacta de la realidad, es sólo una idealización que nos permite tratarla como un problema matemático. Un modelo de un sistema biológico es convertido a sistemas de ecuaciones, aunque la palabra modelo es a menudo usada como el sistema de las ecuaciones correspondientes. La solución de las ecuaciones, ya sea por medios analíticos o numéricos, describe cómo el sistema biológico se comporta ya sea en el tiempo o en equilibrio.

1.1.2 Sistemas de Ecuaciones Diferenciales

Aunque existen muchos tipos de modelos matemáticos diferentes, como los autómatas celulares o las ecuaciones diferenciales en derivadas parciales, han sido los **Sistemas de Ecuaciones Diferenciales** (SED) los de más amplio uso en la modelación de problemas biológicos, debido fundamentalmente, a las facilidades de modelación y de simulación que brindan. Con SED se pueden describir gran cantidad de procesos biológicos y los métodos numéricos para la resolución de los mismos son muy conocidos y están disponibles tanto en la literatura como en diversos software.

Para simular un sistema biológico mediante SED es necesario utilizar infraestructuras distribuidas debido a la exigencia de cálculos que supone realizar millones de simulaciones, si se estima que una simulación demora un segundo en realizarse, tardaría meses en completarse todas las simulaciones en una PC. Sin embargo si aumentamos el número de computadoras a utilizar disminuirá el tiempo requerido para la realización de los cálculos.

1.2 Análisis de Series Temporales

El estudio del comportamiento y evolución temporal de determinada variable, se realizará de forma repetida durante una serie de momentos del tiempo. Esta observación repetida en el tiempo da lugar a una **Serie Temporal**, que pretende describir y predecir el comportamiento de un fenómeno que cambia en el tiempo. El **Análisis de Series Temporales** comprende métodos que ayudan a interpretar los datos, extrayendo información representativa, tanto referente a los orígenes o relaciones subyacentes como a la posibilidad de extrapolar y predecir su comportamiento futuro.

“La forma más sencilla de comenzar el análisis de una Serie Temporal, es mediante su representación gráfica” [4], se dice que es el primer paso obligatorio, ver figura 1. El análisis de una serie implica el manejo conjunto de dos variables, siendo una de ellas nuestra serie temporal y la otra los intervalos o instantes del tiempo sobre los cuales se han realizado las observaciones.

El objetivo del AST es doble. Por un lado se busca explicar las variaciones observadas en la serie en el pasado, tratando de determinar si responden a un determinado patrón de comportamiento. Y por otro, si se consigue definir ese patrón o modelo, se intentará predecir el comportamiento futuro de la misma.

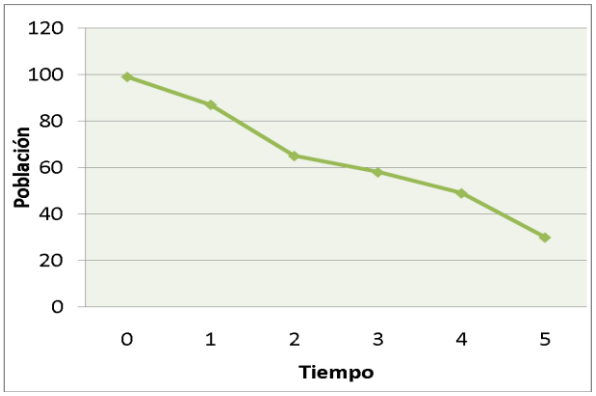


Figura 1. Serie Temporal para una población X**.**

Existen varias formas de analizar Series Temporales, por ejemplo: La Metodología Tradicional y los Modelos ARIMA o Box-Jenkins.

La **Metodología Tradicional** para el AST es bastante sencilla de comprender, y fundamentalmente se basa en descomponer la serie en varias partes:

Tendencia (T). De forma amplia podemos definir la tendencia como aquella componente que recoge el comportamiento de la serie a largo plazo. Mediante la tendencia se puede ver si la serie es estacionaria o constante, de naturaleza lineal, parabólica, exponencial, logística, etc.

Variaciones Cíclicas (C). Es una componente de la serie que recoge oscilaciones periódicas de amplitud superior a un intervalo de tiempo prefijado.

Variaciones Estacionales (E). Es una componente de la serie que recoge oscilaciones que se producen alrededor de la tendencia, de forma repetitiva y en períodos iguales o inferiores a determinado periodo de tiempo.

Variaciones Accidentales (A). Es una componente de la serie que recoge movimientos provocados por factores imprevisibles. También reciben el nombre de variaciones irregulares, residuales o erráticas.

ARIMA es un modelo creado por los profesores Box y G.M. Jenkins en uno de sus trabajos realizados sobre el comportamiento de la contaminación, con el propósito de establecer mejores mecanismos de pronóstico y control. El libro en el que describen la metodología, se convirtió rápidamente en un clásico, y sus procedimientos se utilizan ampliamente desde entonces en diferentes ramas de la ciencia. [5]

Estas formas de realizar el AST son muy factibles pero en realidad lo que interesa es la serie como tal, y ver solamente su comportamiento a lo largo del tiempo. Uno de los usos más habituales de las series de datos temporales es su análisis para predicción y pronóstico, es decir, lo mismo que se había explicado anteriormente cuando se explicaba su concepto. En BioSyS se hacen simulaciones, almacenando de cada una, la Serie Temporal completa, es decir, los valores de cada variable en cada intervalo de tiempo en que se realizó la integración.

1.3 Software que realizan Análisis de Series Temporales

Entre los software que utilizan Análisis de Series Temporales están: STATISTIX, MINITAB, LIMDEP, STATA, SAS, SPSS, BMDP, entre otros. Estos son paquetes estadísticos, que conceptualmente se refieren a un conjunto de programas informáticos específicamente diseñados para el análisis estadístico de datos con el objetivo de resolver problemas de estadística descriptiva, inferencial o ambos.

BMDP es uno de los paquetes de software estadísticos más antiguos. Cubre un amplio abanico de métodos estadísticos (entre ellos las Series Temporales) pero su capacidad para manejar datos es limitada. Sus programas se ejecutan por separado: solo puede accederse a uno de ellos en cada ejecución. “El BMDP clásico no funciona con ventanas, menús y cuadros de diálogo. El usuario principiante considera esto un inconveniente, y efectivamente lo es, en el sentido de que requiere un poco más de atención para aprender su manejo” [6].

Statistical Analysis System (SAS): Comprende amplias posibilidades de procedimientos estadísticos y contiene potentes posibilidades gráficas. Es rentable, ya que permite ejecutar mayor número de procedimientos estadísticos y operativos. Ofrece la mayor flexibilidad para personalizar el manejo y análisis de datos, sin embargo su principal inconveniente es que no resulta fácil aprender a usarlo.

Statistical Package for the Social Sciencies (SPSS): Es un programa estadístico muy potente. Su uso se ha extendido enormemente en casi todos los ámbitos del que hacer científico, por su facilidad de uso, claridad, rapidez, buenas ayudas y por su estabilidad. En general, es un buen software. No

obstante, tiene o más bien implica, algunas limitaciones, como por ejemplo que, es poco flexible, ya que no cuenta con muchos procedimientos de análisis y esto es un serio problema para quienes requieren modelar los datos con rigor; no está pensado para un exclusivo club de estadísticos o investigadores, la pretensión es convertirlo en un producto masivo, lo que implica que está hecho para dummies, que puede ser entendido de dos modos: para novatos o aprendices, o bien, para tontos o lentos. Además, el programa en si, no sorprendería mucho, su interface es más bien tosca, y sus gráficos estéticamente horrorosos.

Minitab: El Software de estadística de Minitab es reconocido por su insuperable facilidad de uso, confiabilidad y amplias capacidades. Es el paquete ideal para enseñar estadística e implementar, porque posee una interfaz lógica que complementa la manera en la que la gente aprende y trabaja y contiene completas capacidades de administración de datos, con poderosas ventanas de importación y exportación de archivos y manipulación de datos, entre otras cosas. Pero no contempla tantos métodos estadísticos como SPSS, SAS o BMDP.

El uso de los paquetes estadísticos tiene algunas limitantes y una de las más importantes es el Costo. Desafortunadamente, el coste de sus licencias está fuera del alcance de la mayor parte de los usuarios y no están diseñados para manejar y analizar grandes cantidades de datos.

Por lo antes planteado, se decidió no utilizar ningún paquete estadístico sino trabajar con **BioSyS**, software creado para satisfacer nuestras propias necesidades y que realiza además, un estudio exhaustivo de los resultados, incorporando varios métodos de Análisis, sin dejar de mencionar que brinda facilidades para la modelación de sistemas biológicos, permitiendo de una forma más sencilla para el investigador, el trabajo con las simulaciones.

1.4 Métodos para el Análisis en BioSyS

En la primera versión del software BioSyS se utilizaron diferentes métodos, que a continuación se explicaran. Aunque estos no se han implementado de forma detallada en la primera versión del Software, la implementación se hizo más bien con el objetivo de demostrar su utilidad en este tipo de estudios.

1.4.1 Dinámica de Poblaciones

Este análisis es muy utilizado cuando se quiere ver si el modelo que se ha construido se ajusta a la realidad del problema en estudio. Para ello se le asignan valores a las variables para los cuales se

conoce el resultado. Si la salida observada se aproxima al resultado esperado entonces el modelo se ajusta al problema real y puede ser utilizado. De lo contrario habría que redefinir el mismo. El análisis de las dinámicas de las poblaciones permite a los investigadores llevarse una idea general del comportamiento del sistema. [7]

1.4.2 Clustering

Es poco probable que se pueda explorar de forma manual toda la información que de cada modelo se almacenada en la BD. Es por ello que se hace necesario el uso de herramientas que permitan hacer minería de estos datos y faciliten la comprensión del sistema en estudio. Una de las técnicas de minería incorporadas a la aplicación propuesta son los algoritmos de clustering. Estos algoritmos permiten agrupar los vectores formados por los valores finales de cada simulación de acuerdo a la distancia existente entre ellos. El usuario puede seleccionar además el algoritmo que desea utilizar, en este caso *k-Means* o *X-Means*. Una vez definida esto el sistema se encarga de realizar el agrupamiento y mostrar el mismo en forma de salida gráfica. El número de clusters que se obtienen se puede asociar con la cantidad de comportamientos diferentes a los que tiende el sistema. [7]

1.4.3 Clasificación

Otro análisis implementado dentro del sistema es el de clasificación. Este consiste en clasificar simulaciones desconocidas a partir de modelos generados con simulaciones previamente clasificadas.

Esta clasificación previa se ha concebido de tres formas. La primera de ellas es tomar para la creación de los modelos aquellos ficheros que salen del análisis de clusters, pues ya en este tipo de análisis se ha hecho una clasificación. Las otras dos formas consisten en que el usuario explora la base de datos y va clasificando las simulaciones de acuerdo a clases definidas por el mismo. La diferencia está en que puede explorar la base de datos manualmente o pedirle al sistema que le muestre *n* dinámicas de forma aleatoria. [7]

1.4.4 Análisis por Reglas

El último análisis implementado dentro de BioSyS se ha llamado análisis por reglas. Este análisis permite crear una especie de gráfica de bifurcaciones donde los estados cualitativamente diferentes se describen mediante reglas lógicas. La idea es que, una vez encontrados estos comportamientos, ya

sea usando análisis de clusters o clasificaciones, el usuario puede estudiar hacia que comportamiento tiende el sistema cuando se varían determinados parámetros dos a dos. [7]

1.5 Técnicas o Algoritmos de Minería de Datos

La **Minería de Datos (MD)** no es más que el análisis, extracción de información y búsqueda de patrones de comportamiento que permanecen ocultos entre grandes cantidades de información, es decir, implícitos en las Bases Datos. Esta técnica la utilizamos para evitar que los investigadores tengan que explorar de forma manual toda la información que esta almacenada en la base de datos. En nuestro trabajo de diploma solo se utilizaran los algoritmos de clustering y clasificación, que están incluidos en el software Weka, imprescindibles para los métodos de análisis que implementamos.

1.5.1 Algoritmos de Clustering

Clustering es el proceso de particionar un conjunto de datos en un conjunto de subclases significativas llamadas grupos, donde se dice que un grupo es una colección de objetos de datos que son similares a otros (agrupar por semejanzas). Los **algoritmos de clustering** se utilizan para identificar estructuras o subclases de los objetos en las bases de datos. La ventaja principal de esta técnica es que las estructuras o los agrupamientos interesantes se pueden encontrar directamente en los datos sin tener ningún conocimiento de fondo. Este concepto es de forma general, pero en BioSyS, teniendo en cuenta la cantidad de información que se maneja, utilizamos los algoritmos de Clustering porque permiten agrupar los vectores formados por los valores finales de cada simulación.

1.5.1.1 SimpleK-Means

Este algoritmo de clustering que utilizaremos en el desarrollo de nuestro trabajo debe definir el número de clusters que se desean obtener, así se convierte en un algoritmo voraz para particionar [8]. A pesar de que presenta algunas debilidades es el que más se adecua a lo que necesitamos.

Los pasos básicos para aplicar el algoritmo son muy simples. Primeramente se determinan la cantidad de clusters en los que se quiere agrupar la información, en este caso las simulaciones. Luego se asume de forma aleatoria los centros por cada clusters. Una vez encontrados los primeros centroides el algoritmo hará los tres pasos siguientes:

1. Determina las coordenadas del centroide.

2. Determina la distancia de cada objeto a los centroides.
3. Agrupa los objetos basados en la menor distancia.

Finalmente quedaran agrupados por clusters, los grupos de simulaciones según la cantidad de clusters que el investigador definió en el momento de ejecutar el algoritmo.

1.5.1.2 X-Means

Este algoritmo, **X-Means** es una variante mejorada del algoritmo K-Means. Su ventaja fundamental está en haber solucionado una de las mayores deficiencias presentadas en K-Means, el hecho de tener que seleccionar a priori el número de clusters que se deseen obtener, a X-Means se le define un límite inferior *K*-min *(*número mínimo de clusters*)* y un límite superior K-max *(*número máximo de clusters*)* y este algoritmo es capaz de obtener en ese rango el número óptimo de clusters, dando de esta manera más flexibilidad al usuario. Durante este proceso, el conjunto de centroides que alcanzan el mejor valor son almacenados, y estos serían la salida final, es decir, los valores finales de cada simulación de acuerdo a la distancia entre ellos. Los mismos son aplicables cuando en la Base de Datos existen al menos 2 simulaciones para el modelo (que son ecuaciones formadas por arreglos de parámetros y condiciones iniciales). Además se ha comprobado que sus resultados son más fiables que los obtenidos con el K-Means, debido a que presenta un valor de distorsión menor, son mucho mejor para realizar Clusters de un conjunto grande de datos y es incluso una variante mucho más rápida.

1.5.1.3 CobWeb

Uno de los algoritmos que pertenecen a la familia de algoritmos jerárquicos es el **CobWeb**, el cual caracteriza por la utilización de aprendizaje incremental, esto quiere decir, que realiza las agrupaciones instancia a instancia. Durante la ejecución del algoritmo se forma un árbol (árbol de clasificación) donde las hojas representan los segmentos y el nodo raíz engloba por completo el conjunto de datos. Al principio, el árbol consiste en un único nodo raíz. Las instancias se van añadiendo una a una y el árbol se va actualizando en cada paso. La clave para saber cómo y dónde se debe actualizar el árbol la proporciona una medida denominada utilidad de categoría, que mide la calidad general de una partición de instancias en un segmento.

1.5.2 Algoritmos de Clasificación

Algoritmo es un conjunto de pasos que nos permite obtener un dato, un resultado; y la **clasificación** no es más que la acción o el efecto de ordenar por clases o categorías, según las propiedades del objeto o concepto en cuestión. Ahora bien, en nuestro trabajo estaremos utilizando el software Weka, que denomina clasificador a cualquier modelo con capacidad de predecir un valor nominal. Para la clasificación se construye un modelo que permite predecir la categoría de las instancias en función de una serie de atributos de entrada. En conclusión, los **algoritmos de clasificación** ordenan por clases o categorías según las propiedades del objeto, siguiendo una serie de pasos que permita obtener un resultado.

1.5.2.1 Árboles de decisión

Los árboles de decisión son los más utilizados para la clasificación y son aquellos cuyos nodos implican comprobar cierto atributo para continuar por una u otra de sus ramas y cuyas hojas proporcionan una clasificación que se aplica a todas las instancias que llegan a esa hoja del árbol. También un árbol de decisión es una estructura en la cual los datos son divididos de acuerdo a un criterio. El Weka posee una serie de algoritmos de aprendizaje que pueden ser utilizados por el usuario a conveniencia, buscando aquellos que generen los modelos más precisos. Para el análisis se ha utilizado el algoritmo J48, que pertenece a los árboles de decisión que incluye la herramienta Weka. El J48 es una variante de los algoritmos ID3 y además una implementación del algoritmo C4.5, uno de los algoritmos de minería de datos más utilizado. En general, este tipo de algoritmos se caracterizan porque son entrenados en una fase de aprendizaje en la que aprenden a modelar el comportamiento de los datos.

1.6 Tendencia y Técnica de los Roles

1.6.1 Roles

Los roles que desempeñaremos son el Analista y el Desarrollador.

- **Analista**: Las personas en este rol representan al cliente y los usuarios finales involucrados, obteniendo información desde los stakeholders para entender el problema a ser resuelto y capturar y ajustar las prioridades para los requerimientos. Este rol puede ser asignado en equipos ágiles y pequeños, es frecuentemente compartido entre varios miembros del equipo

que también desempeñan otros roles, y además, uno o más miembros del equipo desempeñan este rol exclusivamente. Esta alternativa es comúnmente adoptada cuando los requerimientos complejos son difíciles de capturar. [9]

- **Desarrollador**: La persona en este rol es responsable por desarrollar una parte del sistema, incluyendo diseñar esta, para que se ajuste a la arquitectura, posiblemente prototipar la interfaz de usuario y entonces implementar, hacer pruebas unitarias e integrar los componentes que son parte de la solución. [9] Una persona que desempeñe este papel puede tener conocimientos especializados en una determinada área técnica, pero también deben tener una amplia comprensión de todas las tecnologías que intervienen para poder trabajar con otros miembros del equipo técnico.

1.6.2 Artefactos

La metodología de desarrollo OpenUP propone una lista de roles, tareas y artefactos [Anexo 1], pero no todos serán utilizados en el desarrollo del presente trabajo, a continuación se explicarán los roles y artefactos que realmente son de nuestro interés. Los roles que desempeñamos son el Analista y el Desarrollador, que son personas que deben estar bien preparadas para poder realizar cada uno de los artefactos que le corresponden, por ejemplo, deben tener experiencia en el entendimiento del problema, capacidad para crear el modelo visual del sistema y de colaborar eficazmente con el equipo de colaboración, conocimiento del negocio y la tecnología, buenas habilidades, saber definir y crear soluciones técnicas, comprender y ajustarse a la arquitectura, identificar y construir pruebas, además de comunicar el diseño en una forma que otros miembros del equipo puedan entender.

Analista realiza:

1. Glosario: Este artefacto define términos importantes usados por el proyecto. Estos términos son las bases para una colaboración efectiva con los stakeholders y otros miembros del equipo.
2. Requerimientos de Soporte: Este artefacto captura requisitos generales del sistema no capturados en los escenarios o casos de uso, incluyendo requisitos sobre atributos de calidad y requisitos funcionales globales. Esta tarea describe cómo encontrar y resumir los requisitos para el sistema de manera que el alcance del trabajo puede ser determinado.
3. Modelo de Casos de Uso: Este artefacto es la base para el acuerdo entre las partes interesadas (cliente) y el equipo del proyecto (desarrolladores), en relación con la funcionalidad

para el sistema. Además, ayuda a orientar las diversas tareas en el ciclo de vida de desarrollo de software y capta un modelo del sistema de funciones y su entorno.

4. Casos de Uso: Este artefacto captura la secuencia de acciones que un sistema realiza, que produce un resultado observable de valor a su interacción con el sistema. Los casos de uso describen los requisitos de comportamiento para el sistema, y los diferentes usuarios se benefician de diversas maneras:
    - Los Clientes los usan para describir, o aprobar, la descripción del comportamiento del sistema.
    - Los Usuarios Potenciales los usan para entender el comportamiento del sistema.
    - Los Desarrolladores los usan para entender los comportamientos requeridos del sistema de tal manera que ellos puedan identificar clases desde el flujo de eventos de los Casos de Uso.

Desarrollador realiza:

1. Build: Este artefacto es una versión operativa de un sistema o parte de un sistema que demuestra un subconjunto de las capacidades que se incluirá en el producto final. Esta versión ejecutable del sistema suele tener un número de archivos de apoyo que también se consideran parte de este artefacto compuesto. En un ciclo de vida iterativo, cada iteración debe evolucionar la construcción de la iteración anterior a construir, añadiendo más funcionalidad y mejorar la calidad.
2. Diseño**:** Este artefacto describe la realización de la funcionalidad necesaria del sistema en términos de componentes y sirve como una abstracción del código fuente. El objetivo es describir los elementos del sistema a fin de que puedan ser examinados y entendido en un sentido que no sea posible la lectura del código fuente. Se comunica abstracciones particulares de parte de la aplicación y puede describir un encapsulado subsistema, un alto nivel de análisis del sistema, un punto de vista del sistema en un solo contexto, u otras perspectivas que explican una solución a un problema específico que necesita ser comunicad.
3. Implementación: Este artefacto es la colección de uno o más de estos elementos: archivos de código fuente, archivos de datos, crear scripts y el resto de archivos que se transforman en el sistema ejecutable. El objetivo de este artefacto es representar las partes física que componen el sistema a ser construido, organizado de una manera que sea comprensible y manejable.

1.6.3 Patrones

Buschman (1996) define Patrón como una regla que consta de tres partes, la cual expresa una relación entre un Contexto, un Problema y la Solución. Un Patrón sigue el siguiente esquema:

- Contexto: Es una situación de diseño en la que aparece un problema de diseño.
- Problema: Es un conjunto de fuerzas que aparecen repetidamente en el contexto.
- Solución: Es una configuración que equilibra la estructura con componentes y relaciones con el comportamiento a tiempo de ejecución. [10]

1.6.3.1 Patrones de Arquitectura

Partiendo de la definición de Patrón, Buschmann propone los patrones arquitectónicos como descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución. [10] La selección de un patrón arquitectónico es, por lo tanto, una decisión fundamental de diseño en el desarrollo de un sistema de software. Existen muchos patrones de arquitectura, pero aquí sólo mencionaremos algunos de los más interesantes. Los sistemas empresariales distribuidos pueden agrupar los siguientes estilos arquitectónicos:

- Modelo-Vista-Controlador (MVC)
- Arquitecturas en Capas
- Arquitecturas Orientadas a Objetos
- Arquitecturas Basadas en Componentes
- Arquitecturas Orientadas a Servicios

Para la implementación del Módulo de Análisis, teniendo en cuenta las características de estos patrones, se utilizará de los antes mencionados solamente, el **Modelo-Vista-Controlador**, que se explicará más adelante.

1.6.3.2 Patrones de Diseño

Un Patrón de Diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular

(Buschmann, 1996). [10]. Los patrones de diseño "Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan" [11]. En el desarrollo del Módulo de Análisis se utilizarán los patrones de diseño: GRASP, que se explicarán más adelante.

1.7 Herramientas y Metodología

En la primera versión de BioSyS se define utilizar como IDE de desarrollo NetBeans 5.5.1 y lenguaje de programación Java, como herramienta para realizar Minería de Datos el Software Weka y Visual Paradigm como herramienta Case, empleando el lenguaje de modelado UML y la metodología de desarrollo RUP. En esta nueva versión se sustituirá esta metodología por OpenUP y se empleará una nueva herramienta de graficación, el JFreeChart, con el objetivo de graficar las Dinámicas de Población.

1.7.1 Metodologías

Rational Unified Process (RUP)

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, utilizado por una gran variedad de sistemas de software. El ciclo de vida de RUP se caracteriza por, estar dirigido por Casos de Uso, centrado en la arquitectura y ser iterativo e incremental. El desarrollo del software se divide en 4 fases: Inicio, Elaboración, Construcción y Transición. La utilización del RUP tiene muchos atrayentes, pero que sucede, que hay que tener en cuenta un aspecto muy importante y es el tamaño del proyecto. **BioSyS** es un software de gran tamaño, y ya la ingeniería de software más importante fue elaborada. El mismo está formado por módulos lo que hace que menos desarrolladores se vean involucrados en cada módulo y los roles no se puedan asignar en su totalidad. Esto produce que RUP no sea la metodología ideal, por el exceso de documentación que la misma genera y los roles son muchos para la cantidad de desarrolladores con los que se cuenta para el trabajo en cada módulo.

Por lo antes explicado se decidió utilizar la metodología **Open Unified Process** (Open UP) basada en RUP. Es una metodología ágil que se aplica a proyectos de corta duración, es un software de código abierto diseñado para pequeños equipos de trabajo, generalmente de 3-6 personas y de 3-6 meses de esfuerzo de desarrollo. [12]. El mismo preserva las características esenciales de RUP, como son: estar dirigido por Casos de Uso, centrado en la arquitectura y ser iterativo e incremental. La metodología tiene 4 fases en el ciclo de vida del Proyecto. [Anexo2]

1. Fase de Inicio: Es acerca de entender el alcance y los objetivos del proyecto y obtener suficiente información para confirmar que el proyecto debería continuar - o convencerse de que este no debería continuar. El propósito en esta fase es lograr concurrencia entre todos los stakeholders sobre los objetivos del ciclo de vida para el proyecto. [12]
2. Fase de Elaboración: El propósito de esta fase es establecer la línea base de la arquitectura del sistema y proporcionar una base estable para el gran esfuerzo de desarrollo de la siguiente fase. [12]
3. Fase de Construcción: Se enfoca en diseño, implementación y prueba de las funcionalidades para desarrollar un sistema completo. El propósito de esta fase es completar el desarrollo del sistema basado en la arquitectura. [12]
4. Fase de Transición: El propósito en esta fase es asegurarse que el software está listo para entregarse a los usuarios. [12]

1.7.2 Lenguaje de Programación

Java

Debido a la alta productividad, y que requieren poco tiempo de desarrollo, se utiliza Java frente a otros lenguajes de programación como C y C++. Java, emplea el concepto de maquina virtual y el código que genera no es específico a una plataforma en particular. Este lenguaje trabaja con sus datos como objetos y con interfaces a esos objetos y soporta las características propias del paradigma orientado a objetos como son: abstracción, encapsulación, herencia y polimorfismo.

1.7.3 Herramienta CASE

Visual Paradigm

Se define en el proyecto, utilizar como herramienta CASE el **Visual Paradigm** ya que es una potente herramienta para visualizar y diseñar elementos de software utilizando el lenguaje UML, proporciona a los desarrolladores una plataforma que les permita diseñar un producto con calidad de una forma rápida. Facilita la interoperabilidad con otras herramientas CASE como el Rational Rose y se integra con las siguientes herramientas Java: Eclipse/IBM WebSphere, Jbuilder, NetBeans IDE, Oracle Jdeveloper, BEA Weblogic. Genera documentación para el proyecto en HTML, MS Word y PDF.

Además exporta e importa los diagramas en el estándar XML y como imágenes (ya sea con extensiones jpg o png). Es gratis en su edición Community y es multiplataforma.

1.7.4 Herramienta de Desarrollo

NetBeans 5.5.1

Para el desarrollo de la aplicación fue necesario escoger una herramienta según el lenguaje de programación que se seleccionó, por lo tanto se utiliza el **Netbeans 5.5.1** que no es más que una herramienta para el desarrollo de aplicaciones de escritorio usando Java, es de código abierto de gran éxito con una gran base de usuarios. Además es un proyecto GNU que tiene un excelente diseñador de interfaces integrado, es muy rápido y fácil de usar.

1.7.5 Herramienta para realizar Minería de Datos

Weka

Es una colección de algoritmos de aprendizaje de máquina para tareas de minería de datos. Este software contiene herramientas para el procesamiento preliminar de datos, clasificación, regresión, agrupación, asociación, normas, y de visualización. También es muy apropiada para el desarrollo de nuevos planes de aprendizaje de máquina. Weka es un software programado en Java que está orientado a la extracción de conocimientos desde bases de datos con grandes cantidades de información. La licencia de Weka es GPL, lo que significa que este programa es de libre distribución y difusión. Además, ya que Weka está programado en Java, es independiente de la arquitectura, ya que funciona en cualquier plataforma sobre la que haya una máquina virtual Java disponible [13].

1.7.6 Herramienta de Graficación

JfreeChart

Se decide utilizar esta herramienta para graficar las Dinámicas de Población, pero el objetivo fundamental de su utilización, para posteriores versiones, es poder graficar todos los resultados de los Métodos de Análisis en un mismo componente, ya que puede generar gran variedad de gráficos, entre ellos están: las gráficas de barras, los gráficos circulares, de dispersión, de series de tiempo, y otros. JFreeChart es una librería gráfica, libre y para la plataforma de Java. Está diseñado para su uso en aplicaciones, Applet, Servlets y JSP. Se distribuye con el código fuente completo sujeta a los términos

de la General Public Licence (GNU), que permite al mismo ser utilizado tanto en software propietario como en aplicaciones de software libre.

CONCLUSIONES

En este capítulo se abordaron los temas principales que relacionan el AST. Dando una breve explicación de los conceptos fundamentales, relacionados con las simulaciones, Minería de Datos y los distintos tipos de análisis. Además de las herramientas que se utilizan para el desarrollo de la aplicación en general, resaltando los beneficios que estas nos brindan y las metodologías a seguir que nos posibilitan un proceso controlado y las cuales guían nuestro proceso de desarrollo del software.

CAPÍTULO 2: CARACTERISTICAS DEL SISTEMA

En el presente capítulo se analizarán las características del sistema, aquí se abordarán aspectos relevantes como son: las condiciones o capacidades que el software debe cumplir (requisitos funcionales), así como las propiedades o cualidades que el producto debe tener (Requisitos no funcionales). Además se definirán actores que interactúan con el sistema, los casos de uso del sistema, con sus respectivas descripciones y finalmente la relación existente entre actores y casos de uso (Diagrama de Casos de Uso del Sistema).

2.1 Definición de los Actores

Tabla 1. Definición de los actores del sistema.

Actor	Justificación
Investigador	Es el encargado de conectarse al sistema e interactuar con el mismo para realizar el análisis de los resultados obtenidos de las simulaciones.

2.2 Requerimientos Funcionales

R1 Mostrar Dinámicas de Población.

- R1.1 Mostrar gráficas de Poblaciones.

R2 Realizar Clusters.

- R2.1. Cargar fichero de simulaciones no clasificadas.
- R2.2. Agrupar simulaciones.
- R2.3. Mostrar gráfica de Clusters.
- R2.4. Guardar el Resultado de los Clusters.

R3 Realizar Clasificaciones.

- R3.1. Cargar simulaciones clasificadas.
  - R3.1.1 Generar modelo.
  - R3.1.2 Mostrar resultado en forma de árbol.
- R3.2. Cargar modelo existente.
- R3.3. Cargar simulaciones no clasificadas.

R3.3.1. Clasificar simulaciones a partir de un modelo.

R 4 Definir reglas y grupos de reglas.

R 4.1. Crear grupos de reglas.

R 4.2. Crear reglas correspondientes a un grupo.

R 4.3 Crear ecuación lógica correspondiente a cada regla creada.

2.3 Requerimientos No Funcionales

Software: La aplicación permitirá ser instalada sobre cualquier sistema operativo ya que será multiplataforma solo se necesita la instalación de la máquina virtual de java 1.5 o superior. Si se utilizan las simulaciones distribuidas, es necesario, que esté instalado en cada una de las PCs donde se ejecutará, cualquier distribución del sistema operativo Linux y el cliente Grid que se utilizará para este tipo de simulaciones.

Hardware: Se debe contar con 256 MB de memoria RAM como mínimo, preferentemente 512 MB o más para posibilitar que los programas funcionen de forma rápida. Procesadores Pentium IV ya que están listos para ejecutar gran cantidad de procesos y pueden soportar realizar las simulaciones de forma distribuida. Disco duro de 20 GB como mínimo en dependencia de la cantidad de información a almacenar, que generalmente será considerable debido a que cada modelo genera un gran número de simulaciones y estas necesitan ser almacenas para posteriormente ser tratadas por los distintos métodos de análisis.

Requerimientos del diseño y la implementación: Lenguaje de programación Java. NetBeans 5.5.1, como IDE de desarrollo. Visual Paradigm como herramienta CASE.

Usabilidad: El sistema ofrecerá al investigador la posibilidad de realizar el análisis de las series temporales generadas en las simulaciones por diferentes métodos. Dándole al investigador la posibilidad de escoger el método que desee según las características que presenta el diseño de la aplicación con sus interfaces correspondientes que ayudan al investigador a navegar por la aplicación de una forma sencilla.

Rendimiento: El cálculo distribuido permitirá que la aplicación tenga un alto rendimiento, ya que este agiliza la obtención de resultados de los métodos de análisis aplicados en el Software BioSyS.

Soporte: Se brindaran los servicios de instalación del software y configuración del mismo. Se le dará un seguimiento a la aplicación para ir mejorándola progresivamente. La Aplicación será portable, pero exige personal capacitado, debido a su alta complejidad de instalación y configuración.

Portabilidad: El sistema funcionará en cualquier Sistema Operativo que cuente con la maquina virtual de java 1.5 o superior.

Seguridad: Este es quizás el tipo de requerimiento más difícil, que provocará los mayores riesgos si no se maneja correctamente. La seguridad puede ser tratada en tres aspectos diferentes:

- Confidencialidad: La información manejada por el sistema está protegida de acceso no autorizado y divulgación.
- Integridad: la información manejada por el sistema será objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos. Pueden incluir también mecanismos de chequeo de integridad y realización de auditorías.
- Disponibilidad: Significa que los usuarios autorizados se les garantizará el acceso a la información y que los dispositivos o mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

La seguridad de la Aplicación será implementada cuando se integre el Módulo de Análisis al software BioSyS y las funcionalidades que brinda el mismo solo serán accesibles a los usuarios que cuenten con los privilegios requeridos.

Políticos-culturales: El sistema puede estar expuesto a caer en manos de alguna persona que pueda darle un uso indebido. También son importantes los obstáculos que se presentan al querer patentar como país bloqueado que somos, las licencias de los Software que desarrollamos.

Legales: El sistema está desarrollado por el Grupo de Bioinformática de la Universidad de las Ciencias Informáticas (UCI), en conjunto con el Centro de Inmunología Molecular (CIM), por tanto los derechos sobre esta Aplicación están solamente permitidos a las personas autorizadas por el Msc. Noel Moreno Lemus quien dirige el Grupo de Bioinformática en la Facultad 6, por tanto se prohíbe su comercialización e instalación sin su debida autorización.

Confiabilidad: El sistema no debe presentar ningún fallo pero en caso de que esto ocurra se minimizará la pérdida de información.

Interfaz: El sistema deberá tener una interfaz externa amigable, que sea sencilla y fácil de entender por el investigador, que le brinde a este una forma escalonada de acceder a los distintos métodos de análisis en interfaces diferentes presentándole así una organización jerárquica evitando que el usuario pierda la orientación dentro de la aplicación.

2.4 Casos de Uso del Sistema

1. Mostrar Dinámicas de Población.
2. Realizar Clusters.
3. Realizar Clasificaciones.
4. Definir reglas y grupos de reglas.

2.5 Diagrama de Casos de Uso del Sistema

El modelo de Casos de Uso describe la funcionalidad propuesta del sistema, para ello muestra la relación entre actor–caso de uso. Un actor es un usuario del sistema (Investigador) y un caso de uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. El siguiente diagrama muestra la interacción entre el investigador y cada caso de uso.

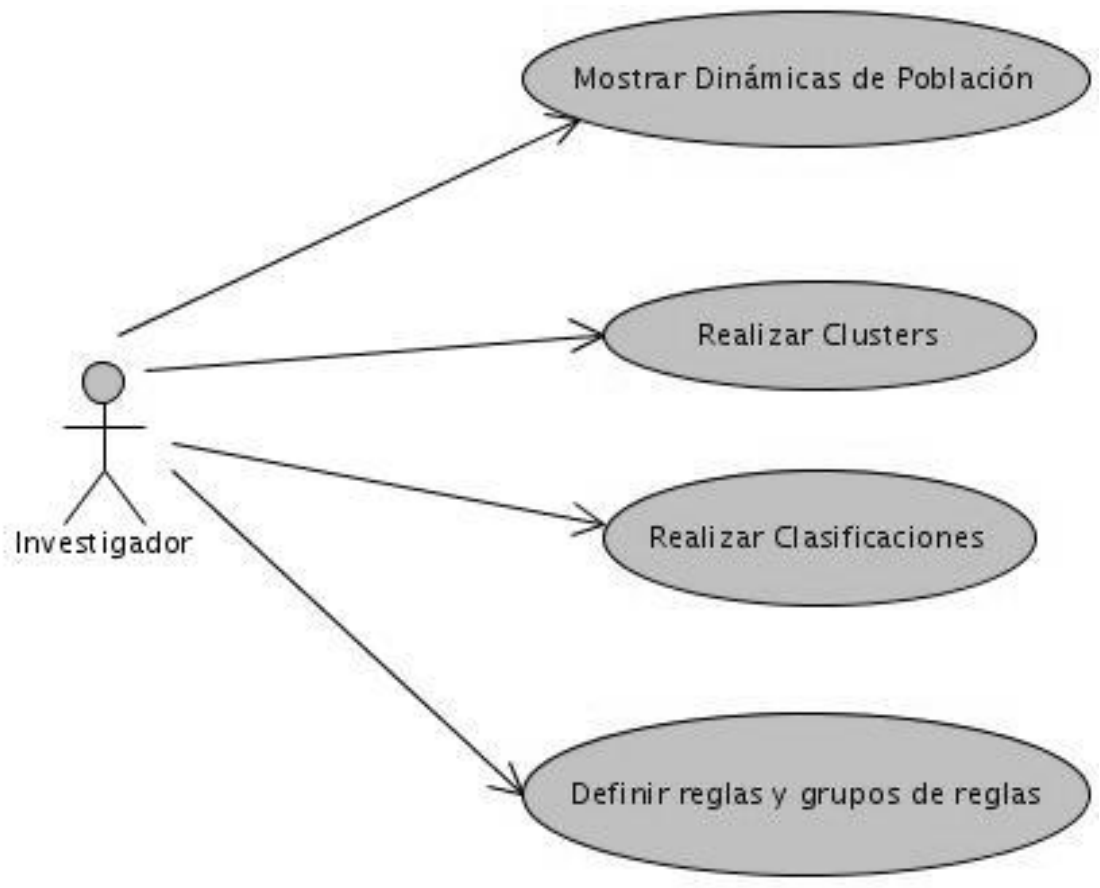


Figura 2. Diagrama de Casos de Uso.

2.6 Descripción de Casos de Uso del Sistema

Tabla 2. Descripción Textual del Caso de Uso Mostrar Dinámicas de Población.

Caso de Uso	**Mostrar Dinámicas de Población**
Actores	Investigador
Propósito	Analizar las salidas gráficas de las simulaciones donde se representan las Series Temporales.

Resumen	El caso de uso se inicia cuando el investigador desea analizar las Dinámicas de Población correspondientes a los resultados obtenidos de un grupo de simulaciones. El sistema muestra como resultado en una gráfica el comportamiento de las poblaciones en diferentes intervalos de tiempo (Dinámica de Población).
Referencias	R1.1
Precondiciones	El investigador debe estar autenticado.
Poscondiciones	Se grafican las Dinámicas de un grupo de simulaciones.
Curso normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. El investigador escoge Mostrar Dinámica de Población.	2. El sistema ejecuta el algoritmo que crea los resultados.
	3. El sistema envía estos resultados a un algoritmo que se encarga de generar la gráfica con la herramienta JFreeChart.
	4. El sistema muestra la gráfica con los resultados de las simulaciones, donde se observa el comportamiento de las Series Temporales por Población.
Prototipo de Interfaz	
Ver figura 18	
Prioridad	Crítico

Tabla 3. Descripción Textual del Caso de Uso Realizar Clusters.

Caso de Uso	**Realizar Clusters**
Actores	Investigador
Propósito	Realizar análisis de las simulaciones que se encuentran en un fichero, agrupándolas por tres tipos de algoritmos de clusters.
Resumen	El caso de uso se inicia cuando el investigador interactúa con la

	aplicación para realizar clusters. Como resultado el sistema muestra una gráfica con las simulaciones agrupadas por clusters según el algoritmo seleccionado.
Referencias	R2.1, R2.2, R2.3, R2.4, R2.5.
Precondiciones	El investigador debe estar autenticado
Poscondiciones	Graficar las simulaciones agrupadas por clusters.
Curso normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. El investigador escoge realizar clusters.	2. El sistema muestra la interfaz para realizar clusters.
3. El investigador indica cargar fichero.	4. El sistema muestra una interfaz con dos opciones.
5. El investigador selecciona: • "Cerrar". • "Abrir".	6. El sistema realiza lo siguiente: • Si el investigador seleccionó "Cerrar", entonces termina el Caso de Uso • Si el investigador seleccionó "Abrir", entonces el sistema le permite cargar el fichero. Se ejecuta la acción 7.
7. El investigador escoge el algoritmo.	8. El sistema muestra los datos que debe introducir según el algoritmo seleccionado.
9. El investigador introduce datos. • Si selecciona el algoritmo "SimpleKMeans" (Introducir la cantidad de clusters) • Si selecciona "XMeans"	

(Introduce rango de clusters mínimo-máximo)	
10. El investigador indica “Clusterizar”.	11. El sistema verifica que el investigador haya seleccionado un algoritmo y que los datos estén correctos.
	12. El sistema muestra la gráfica con las simulaciones agrupadas según el algoritmo seleccionado.
	13. El sistema brinda la posibilidad de guardar los datos de las simulaciones con el cluster al que pertenece.
14. El investigador indica guardar los datos.	15. El sistema brinda la posibilidad de especificar nombre y lugar de lo que desea guardar.
16. El investigador introduce nombre y especifica lugar donde desea guardar.	17. El sistema guarda las simulaciones agrupadas.
Prototipo de Interfaz	
Ver figura 19	
Flujo Alterno	
Acción del actor	**Respuesta del Sistema**
	11.1. Si los datos no están correctos y el investigador no ha escogido ningún algoritmo, el sistema muestra un mensaje de alerta.
14.1. El investigador no desea guardar los datos y termina el caso de uso.	
Prioridad	Crítico

Tabla 4. Descripción Textual del Caso de Uso Realizar Clasificaciones.

Caso de Uso	**Realizar Clasificaciones**
Actores	Investigador
Propósito	Clasificar simulaciones a partir de modelos generados de simulaciones previamente clasificadas.
Resumen	El caso de uso se inicia cuando el investigador interactúa con la aplicación para clasificar simulaciones. Como resultado el sistema muestra una gráfica con las simulaciones clasificadas partiendo de un modelo generado de simulaciones previamente clasificadas.
Referencias	R3.1, R3.1.1, R3.1.2, R3.2, R3.3, R3.3.1
Precondiciones	El investigador debe estar autenticado.
Poscondiciones	Mostrar los resultados de la clasificación en una gráfica de clusters.
Curso normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. El investigador escoge la opción realizar clasificaciones	2. El sistema permite realizar las clasificaciones de dos formas.
3. El investigador selecciona: • "Clasificar Empleando Simulaciones Clasificadas". • "Clasificar Empleando Modelo Generado".	4. El sistema, realiza lo siguiente: • Si el investigador decide "Clasificar Empleando Simulaciones Clasificadas", entonces ir a la Sección "Clasificar Empleando Simulaciones Clasificadas". • Si el investigador decide "Clasificar Empleando Modelo Generado", entonces ir a la Sección "Clasificar Empleando Modelo Generado".
	5. El sistema permite cargar simulaciones no clasificadas.

6. El investigador carga las simulaciones no clasificadas.	
7. El investigador acepta cargar las simulaciones no clasificadas.	8. El sistema permite clasificar.
9. El investigador indica Clasificar Simulaciones.	10. El sistema verifica que los ficheros cargados anteriormente este correctos
	11. El sistema clasifica las simulaciones no clasificadas.
	12. El sistema guarda las simulaciones clasificadas.
	13. El sistema permite al investigador ver las simulaciones clasificadas en una gráfica de clusters.
13. El investigador selecciona “Mostrar el Resultado Clasificado”.	14. El sistema muestra el resultado de las simulaciones clasificadas en una gráfica de clusters.
Prototipo de interfaz	
Ver figura 20 y 21	
Flujo Alterno	
Acción del actor	**Respuesta del Sistema**
3.1. Si el investigador no selecciona ninguna de las dos formas de clasificar, termina el Caso de Uso.	
9.1. Si el investigador no desea clasificar, terina el Caso de Uso.	
	11.1 Si el investigador no carga fichero o los ficheros cargados no son correctos, el sistema muestra un mensaje de alerta.
13.1. Si el investigador	

no selecciona “Mostrar el Resultado Clasificado”, termina el Caso de Uso	
Sección “Cargar Simulaciones Clasificadas”	
Acción del actor	**Respuesta del Sistema**
	1. El sistema muestra la interfaz que permite cargar las simulaciones clasificadas.
2. El investigador carga el fichero con las simulaciones clasificadas.	
3. El investigador indica generar árbol.	4. El sistema genera el árbol.
	5. El sistema muestra el árbol generado con las simulaciones clasificadas.
	6. El sistema permite guardar el modelo y la imagen del árbol.
7. El investigador selecciona: • “Guardar Imagen árbol”. • “Guardar Modelo”.	8. El sistema realiza lo siguiente: • Si el investigador selecciona “Guardar Imagen árbol”, el sistema indica que debe introducir el nombre de la imagen. Se ejecuta la acción 9. • Si el investigador selecciona “Guardar Modelo”, el sistema indica que debe introducir el nombre del mismo. Se ejecuta la acción 13 de esta sección.
9. El investigador introduce el nombre de la imagen del árbol.	
10. El investigador indica Aceptar.	11. El sistema verifica que se haya introducido un nombre de la imagen del árbol.
	12. El sistema guarda la imagen del árbol. Se ejecuta la acción 7.

13. El investigador introduce el nombre del modelo.	
14. El investigador indica Aceptar.	15. El sistema verifica que se haya introducido un nombre para el modelo.
	16. El sistema genera el modelo y lo guarda. Se ejecuta la acción 5 del Flujo Normal de los Eventos.
Flujo Alterno	
Acción del actor	**Respuesta del Sistema**
2.1 Si el investigador no carga el fichero termina el Caso de Uso.	
3.1. Si investigador no desea generar árbol termina el Caso de Uso.	
7.1 Si el investigador no desea ni guardar el modelo, ni la imagen, termina el Caso de Uso.	
	12.1 Si el investigador no introduce el nombre de la imagen del árbol, el sistema muestra un mensaje de alerta.
	16.1. Si el investigador no introduce el nombre del modelo el sistema muestra un mensaje de alerta.
Sección "Cargar Modelo Generado"	
Acción del actor	**Respuesta del Sistema**
	1. El sistema muestra una interfaz que permite al investigador cargar las simulaciones clasificadas.
2. El investigador carga el fichero con	3. El sistema permite al investigador cargar el modelo generado.

simulaciones clasificadas.	
4. El investigador carga el modelo generado.	
5. El investigador acepta cargar el modelo.	6. El sistema ejecuta la acción 5 del Flujo Normal de los Eventos.
Flujo Alterno	
Acción del actor	**Respuesta del Sistema**
2.1. Si el investigador no carga el fichero con simulaciones clasificadas, termina el Caso de Uso.	
4.1. Si el investigador no carga ningún modelo, termina el Caso de Uso.	
Prioridad	Crítico

Tabla 5. Descripción Textual del Caso de Uso Definir reglas y grupos de reglas.

Caso de Uso	**Definir reglas y grupos de reglas**
Actores	Investigador
Propósito	Definir reglas y grupos de reglas, así como las ecuaciones lógicas correspondientes a las mismas.
Resumen	El caso de uso se inicia cuando el investigador decide definir reglas, para el cual debe crear las reglas y los grupos de reglas correspondientes a las mismas, es decir, el nombre del grupo y el nombre de las reglas que pertenecen a ese grupo, así como definir la ecuación lógica perteneciente a cada regla.
Referencias	R 4.1, R 4.2, R 4.3.
Precondiciones	El investigador debe estar autenticado.

Poscondiciones	Guardar los grupos de reglas con la ecuación que le corresponde a cada regla.
Curso normal de los eventos	
Acción del actor	**Respuesta del Sistema**
1. El investigador selecciona la opción Definir reglas y grupos de reglas.	2. El sistema muestra una interfaz para que el investigador pueda crear grupos de reglas.
3. El investigador introduce el nombre del grupo.	
4. El investigador indica crear grupo.	5. El sistema verifica que no existan dos grupos con el mismo nombre.
	6. El sistema muestra el grupo creado.
	7. El sistema permite: • “Crear un grupo de reglas”. • “Crear regla”. • “Terminar”.
8. El investigador puede: • “Crear un grupo de reglas”. • “Crear regla”. • “Terminar”	9. El sistema realiza lo siguiente: • Si el investigador decide “Crear un grupo de reglas” entonces ir a la acción 3. • Si el investigador decide “Crear regla” entonces el sistema permite al investigador entrar los datos. Se ejecuta la acción 10. • Si el investigador decide “Terminar”, entonces se termina el Caso de Uso.
10. El investigador selecciona un grupo para el cual adicionará una regla.	

11. El investigador introduce el nombre de la regla.	12. El sistema permite asignar ecuación lógica a la regla de dos formas.
13. El investigador selecciona: • "Manualmente". • "Cargando Fichero".	14. Si el investigador selecciona: • "Manualmente", el sistema permite introducir la ecuación. Se ejecuta la acción 15. • "Cargando Fichero", el sistema permite cargar el fichero. Se ejecuta la acción 19.
15. El investigador introduce la ecuación.	
16. El investigador indica adicionar regla.	17. El sistema verifica que el nombre de la regla no exista en el mismo grupo y que la ecuación se haya introducido correctamente.
	18. El sistema adiciona la regla. Volver a la acción 7
19. El investigador indica cargar fichero.	20. El sistema muestra una interfaz con dos opciones.
21. El investigador selecciona: • "Cerrar". • "Abrir".	22. El sistema realiza lo siguiente: • Si el investigador seleccionó "Cerrar", entonces termina el Caso de Uso • Si el investigador seleccionó "Abrir", entonces el sistema le permite cargar el fichero. Se ejecuta la acción 23.
23. El investigador selecciona la ecuación e indica "copiar".	24. El sistema muestra un editor de ecuaciones.
25. El investigador indica "pegar" sobre el editor. Se ejecuta la acción 16.	
Prototipo de Interfaz	
Ver figura 22	

Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	6.1. Si el investigador no introduce ningún nombre para el grupo, el sistema muestra un mensaje de alerta.
	18.1. Si el investigador no introduce ningún nombre para la regla, o ya existe una regla con ese nombre, o la ecuación no es correcta, el sistema muestra un mensaje de alerta.
Prioridad	Crítico

CONCLUSIONES

En este capítulo se elaboraron y describieron los principales artefactos que plantea la metodología Open-Up para iniciar el desarrollo del Software, destacando la obtención y descripción detallada de los Casos de Uso del sistema y el Actor involucrado en el proceso. También se exponen las funcionalidades y cualidades que el software debe tener a través de los requerimientos funcionales y no funcionales, para la obtención final de un producto que satisfaga las necesidades de nuestros clientes.

CAPÍTULO 3: DISEÑO DEL SISTEMA

En este capítulo se agrupan los artefactos y descripciones necesarias para que el diseño de la aplicación este guiado por un proceso organizado, que involucra Patrones de Diseño y Arquitectura muy importantes que definen la estructura del software. Finalmente se presentan los Diagramas de Clases del Diseño e Interacción que muestran las clases con sus relaciones y propiedades internas.

3.1 Patrones de Arquitectura

Modelo–Vista-Controlador

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. [14]

Modelo: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada [14]. Este es la representación específica del dominio de la información sobre la cual funciona la aplicación.

Vista: Muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. [14]

Controlador: Este responde a eventos, que son traducidos a solicitudes de servicio para el modelo o la vista, usualmente acciones del usuario (movimientos o pulsación de botones del ratón, pulsaciones de teclas) e invoca cambios en el modelo y probablemente en la vista. El usuario interactúa con el sistema a través de los controladores.[14]

La figura 3 representa un ejemplo de cómo se pone de manifiesto este patrón en el presente trabajo. Observándose la relación entre los tres componentes o áreas en que se propone dividir el MVC y tratándose de agrupar las clases por áreas o paquetes. Como un ejemplo se tiene en la Vista las clases interfaces, CI_Principal y CI_Cargar_Fichero, donde el usuario hará la solicitud que obtendrá el Controlador que en este caso son las clases CC_Dinamica_Reglas, CC_Clusters y CC_Clasificaciones, utilizando o apoyándose del Modelo que agrupa las clases contenedoras de datos, entre ellas CE_Grupos_Reglas, donde se encuentra la información necesaria de los grupos y las reglas, para dar respuesta a la solicitud realizada.

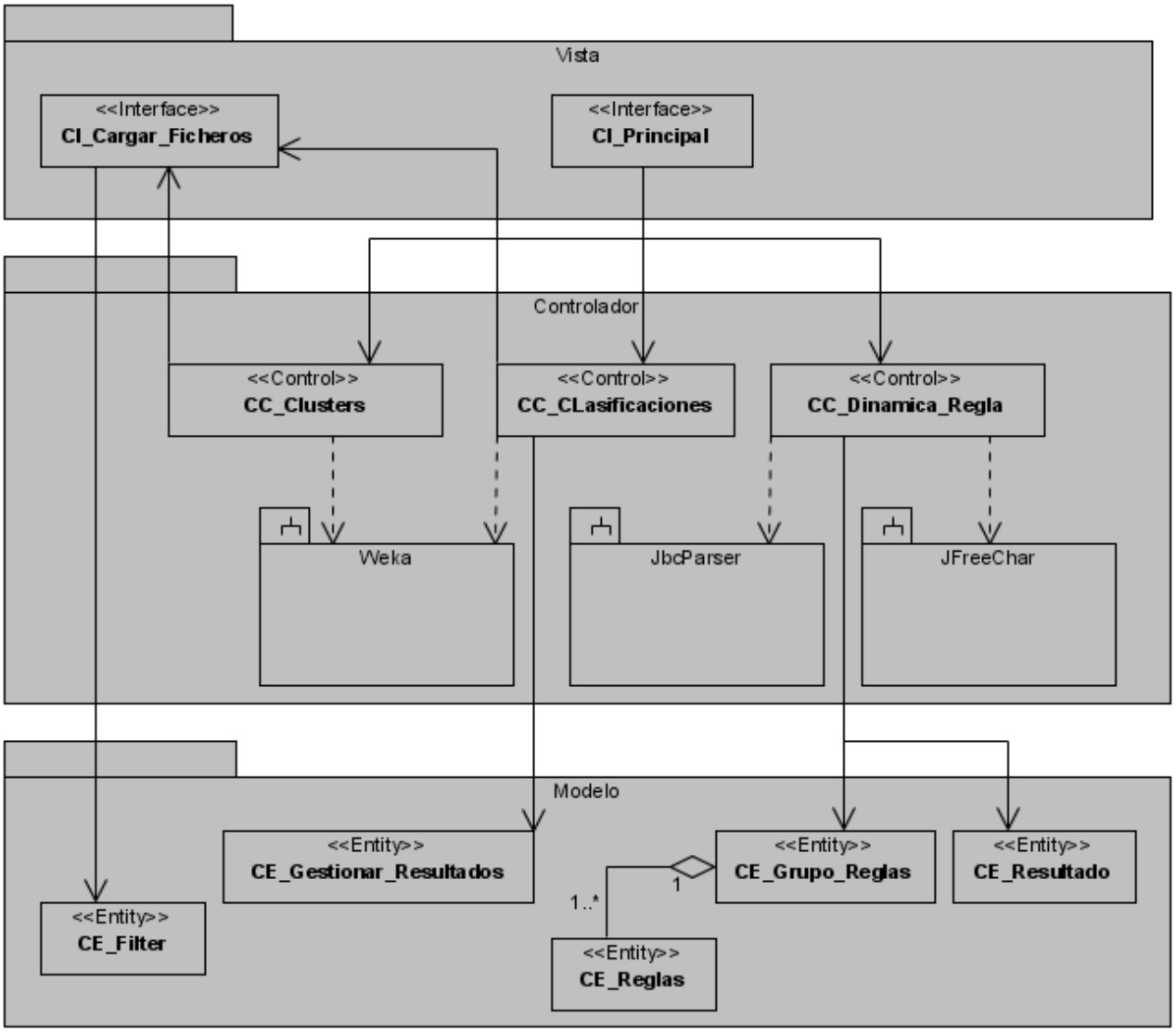


Figura 3. Ejemplo donde se pone de manifiesto el Patrón Modelo-Vista-Controlador**.**

3.2 Patrones de Diseño

GRASP(Patrones Generales de Software para Asignar Responsabilidades)

Para explicar donde se reflejan los Patrones de Diseño en el presente trabajo, se parte de un breve concepto de estos patrones. Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. [15]. Los patrones GRASP, se aplican en la asignación de responsabilidades, estos nos ayudan a encontrar los patrones de diseño, que son más concretos. Los patrones básicos de GRASP son:

1. **Experto**: Asigna responsabilidades a las clases que tienen la información necesaria para cumplir con la responsabilidad. [16]. En la figura 4 se representa la clase controladora CC_Dinamica_Regla que cumple con este patrón, ya que maneja toda la información que contienen los grupos de reglas y a su vez las reglas.
2. **Creador**: Guía la asignación de responsabilidades relacionadas con la creación de objetos. [16]. Este patrón se evidencia en la figura 4 donde la clase controladora es la encargada de crear los grupos de reglas.
3. **Bajo acoplamiento**: Acoplamiento bajo significa tener las clases lo menos ligadas entre sí. [16]. La figura 4 muestra como se pone de manifiesto este patrón ya que las clases se comunican con el menor número de clases posibles, es decir, CC_Dinamica_Regla se relaciona con CE_Grupo_Reglas y esta con CE_Reglas.

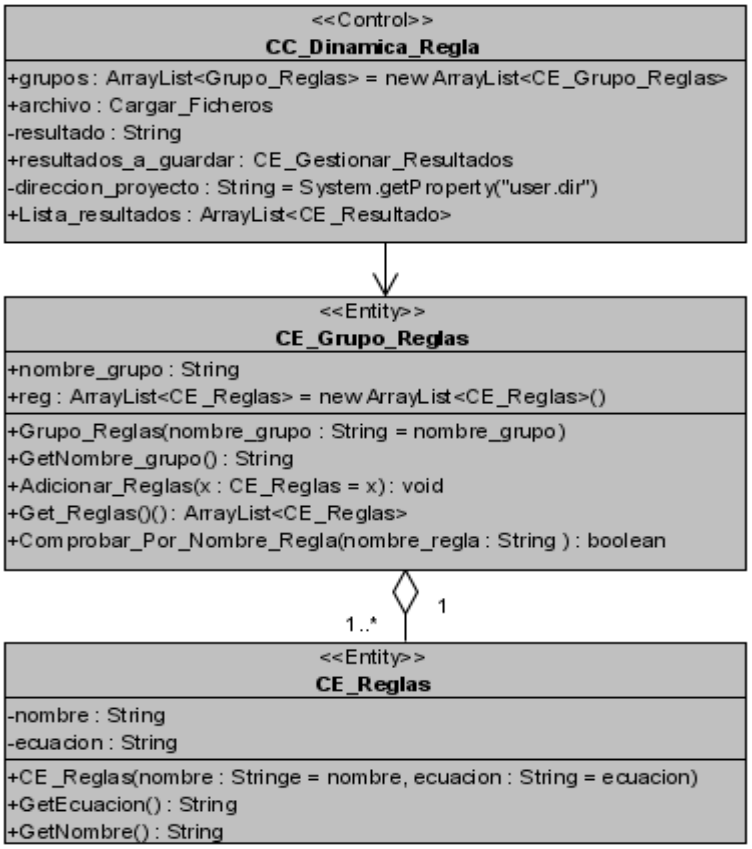


Figura 4. Ejemplo que evidencia los Patrones Experto, Creador y Bajo Acoplamiento.

4. **Alta cohesión**: Mantiene la complejidad dentro de límites manejables. [16]. La figura 5 muestra como se pone de manifiesto este patrón, ya que las clases se reparten las responsabilidades según las tareas que se realizan. Si el investigador desea realizar clusters, la clase CC_Clusters es la encargada de manejar toda la información relacionada con los clusters. Por otro lado si el investigador desea clasificar la clase CC_Clasificaciones se encarga de todo el proceso y si desea trabajar con las reglas y los grupos de reglas la clase CC_Dinamica_reglas, es la responsable.

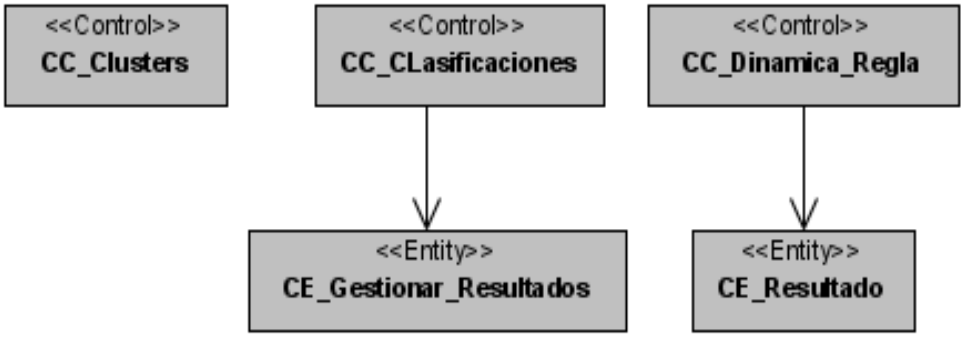


Figura 5. Ejemplo que evidencia el Patrón Alta Cohesión.

5. **Controlador:** Asigna las responsabilidades de capturar los eventos del sistema a las clases. [16]. La figura 6 representa este patrón en la clase CI_Principal, que se encarga de manejar todos los mensajes de los eventos del sistema. investigador.

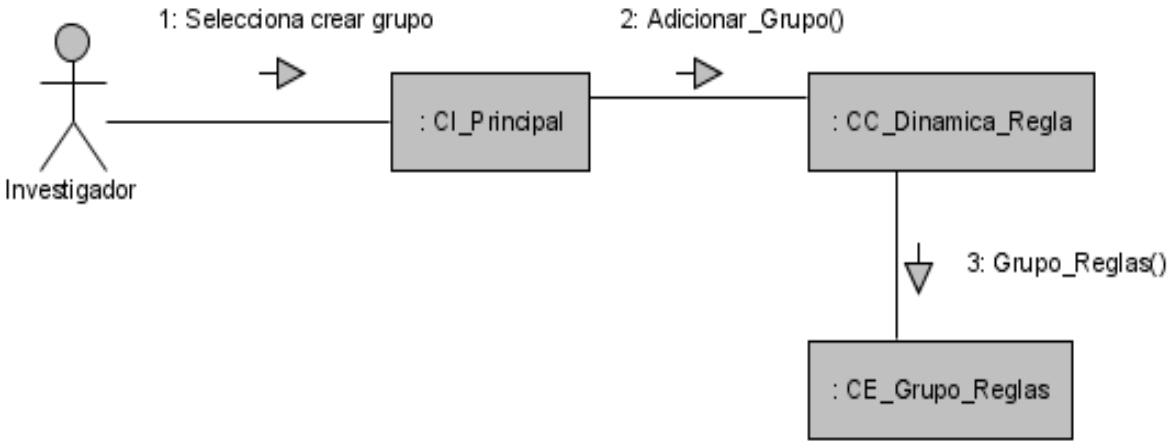


Figura 6. Ejemplo que evidencia el Patrón Controlador.

3.3 Diagrama de Clases

Un Diagrama de Clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases (nombre, atributos y métodos), interfaces, colaboraciones y las relaciones entre ellas. Las figuras 7, 8, 9 y 10 muestran las relaciones entre las clases Interfaces, Controladoras y Entidades. Además se observa la relación entre una clase controladora y un subsistema (parte del sistema que encapsula comportamientos y expone un conjunto de interfaces). En el presente trabajo se utilizan tres subsistemas Weka, JbcParser y el JFreeChart.

El **Subsistema Weka** contiene los paquetes y clases necesarias para que los métodos de clusters y clasificaciones funcionen de forma correcta. Entre los paquetes y clases más importantes que se emplean se encuentran: los paquetes clusterers, classifiers, core y gui que agrupan las clases: Cobweb, X-Means y SimpleKMeans, J48, ArffViewer y Evaluation entre otras, que contienen los métodos útiles para el desarrollo del presente trabajo. [Anexo 3]

El **Subsistema JFreeChart** contiene los paquetes y clases que se utilizan para la graficación de las dinámicas que contienen las series temporales según la cantidad de poblaciones, algunas de estas clases son: CharFactory, XYSeries, ChartPanel, Plot y JfreeChart.

El **Subsistema JbcParser** contiene los paquetes y clases que se utilizan para evaluar las ecuaciones lógicas que introduce el investigador en el editor de ecuaciones y la clase más importante que utiliza es, IMathParser que se encarga fundamentalmente de comprobar, asignar valores, crear variables y hacer cálculos matemáticos.

A continuación se muestran los diferentes Diagramas de Clases del Diseño por Caso de Uso donde se evidencia lo explicado anteriormente.

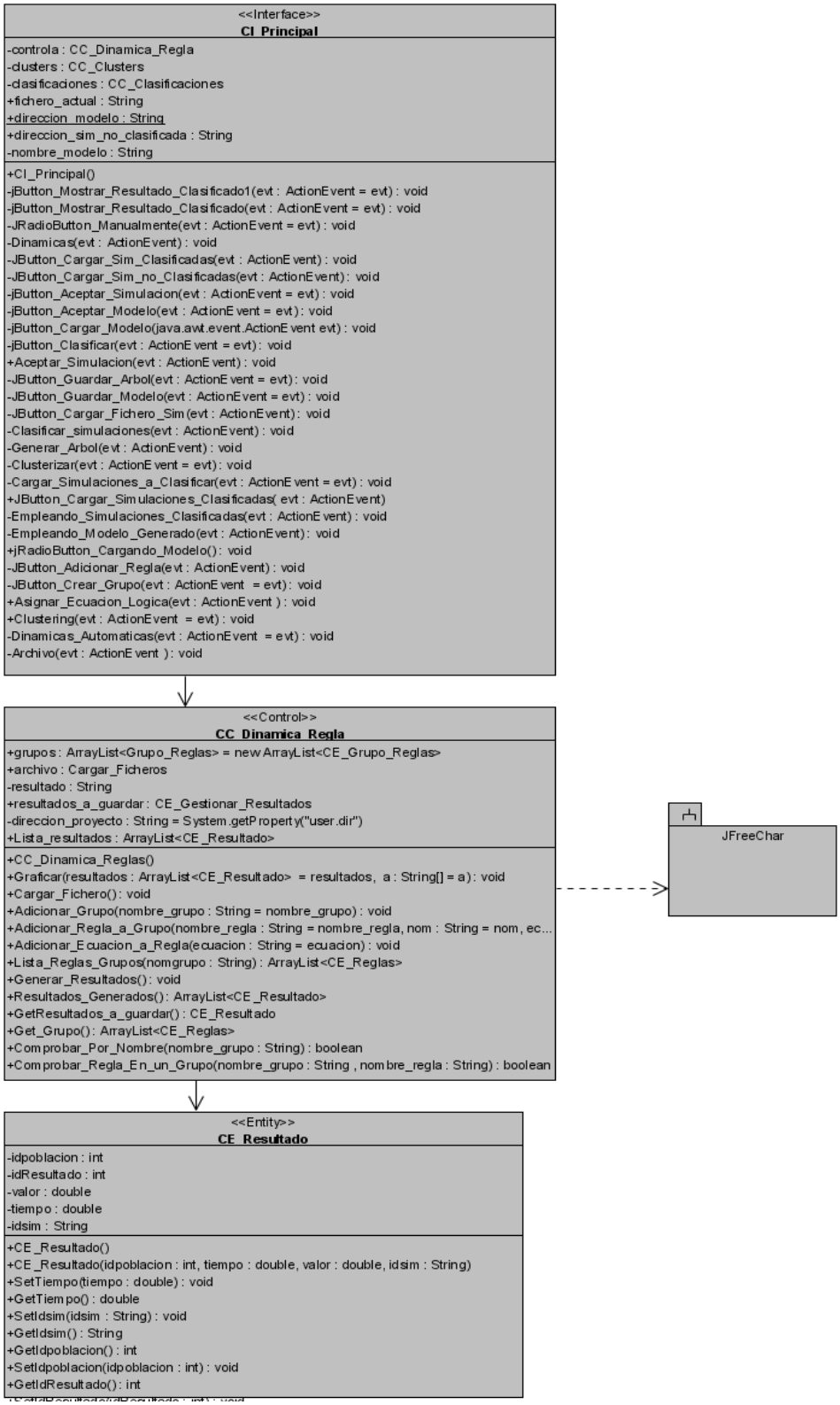


Figura 7. Diagrama de Clases del Diseño del CU “Mostrar Dinámicas de Población”.

Figura 8. Diagrama de Clases del Diseño del CU "Realizar Clusters"**.**

Figura 9. Diagrama de Clases del Diseño del CU “Realizar Clasificaciones”.

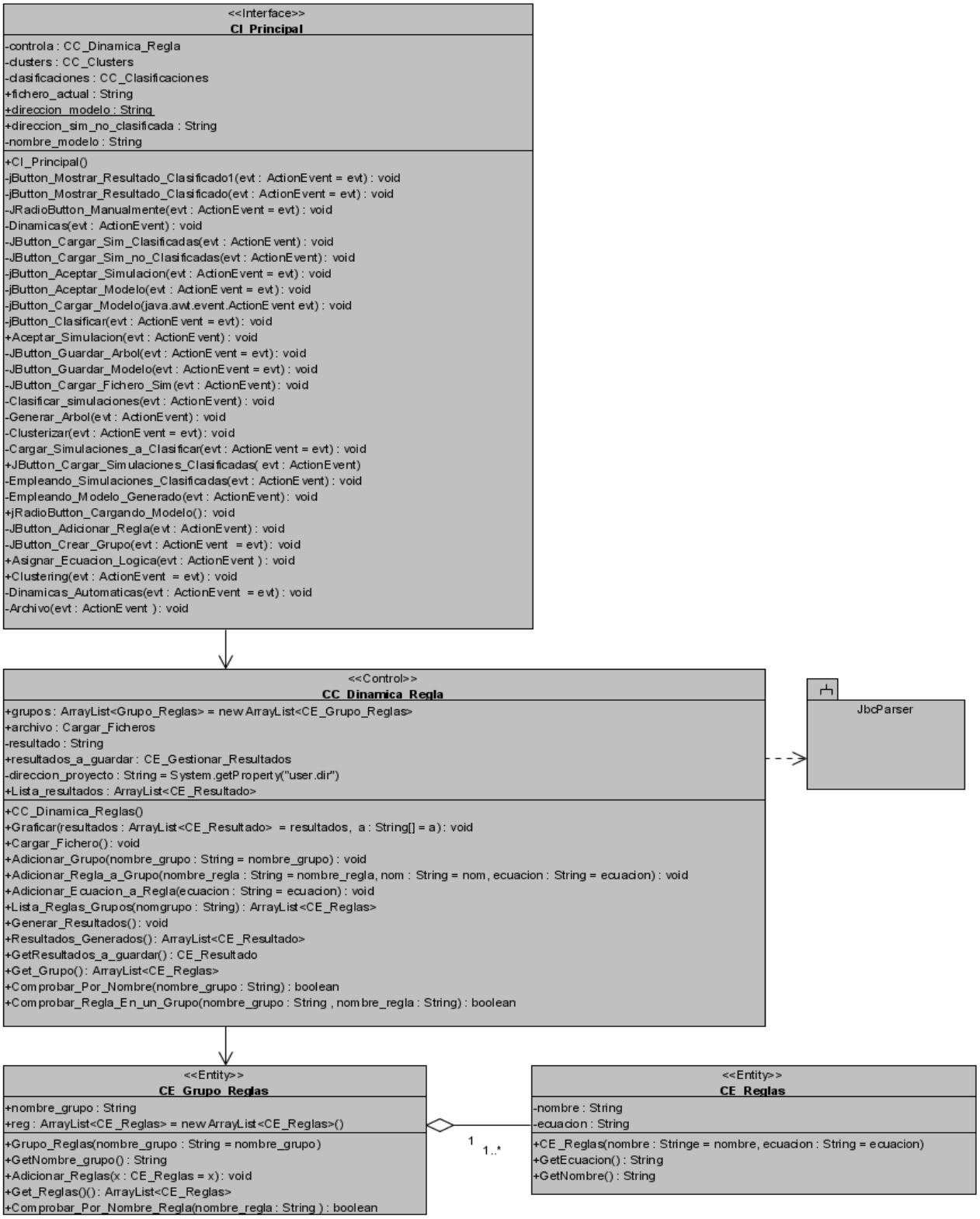


Figura 10. Diagrama de Clases del Diseño del CU “Definir reglas y grupos de reglas”.

3.4 Diagrama de Interacción

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema, lo que conlleva a modelar instancias concretas o prototípicas de clases interfaces, componentes y nodos, junto con los mensajes enviados entre ellos, todo en el contexto de un escenario que ilustra un comportamiento. Cada diagrama será una visión gráfica de un escenario. Existen dos tipos de diagramas de interacción: Diagramas de Secuencia (que destacan la ordenación temporal de los mensajes) y Diagramas de Colaboración (que destaca la organización estructural de los objetos que envían y reciben mensajes). A continuación se muestran los Diagramas de Secuencia.

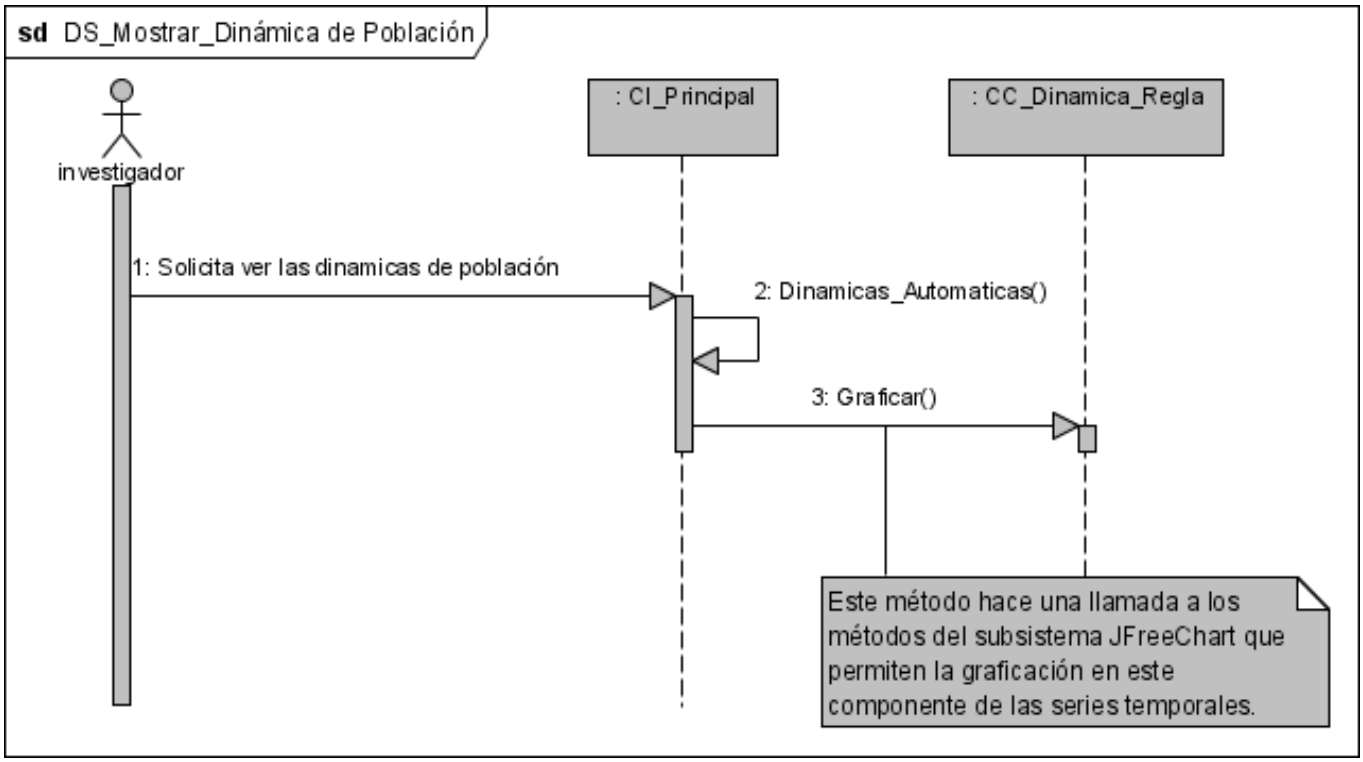


Figura 11. Diagrama de Secuencia de “Mostrar Dinámicas de Población”.

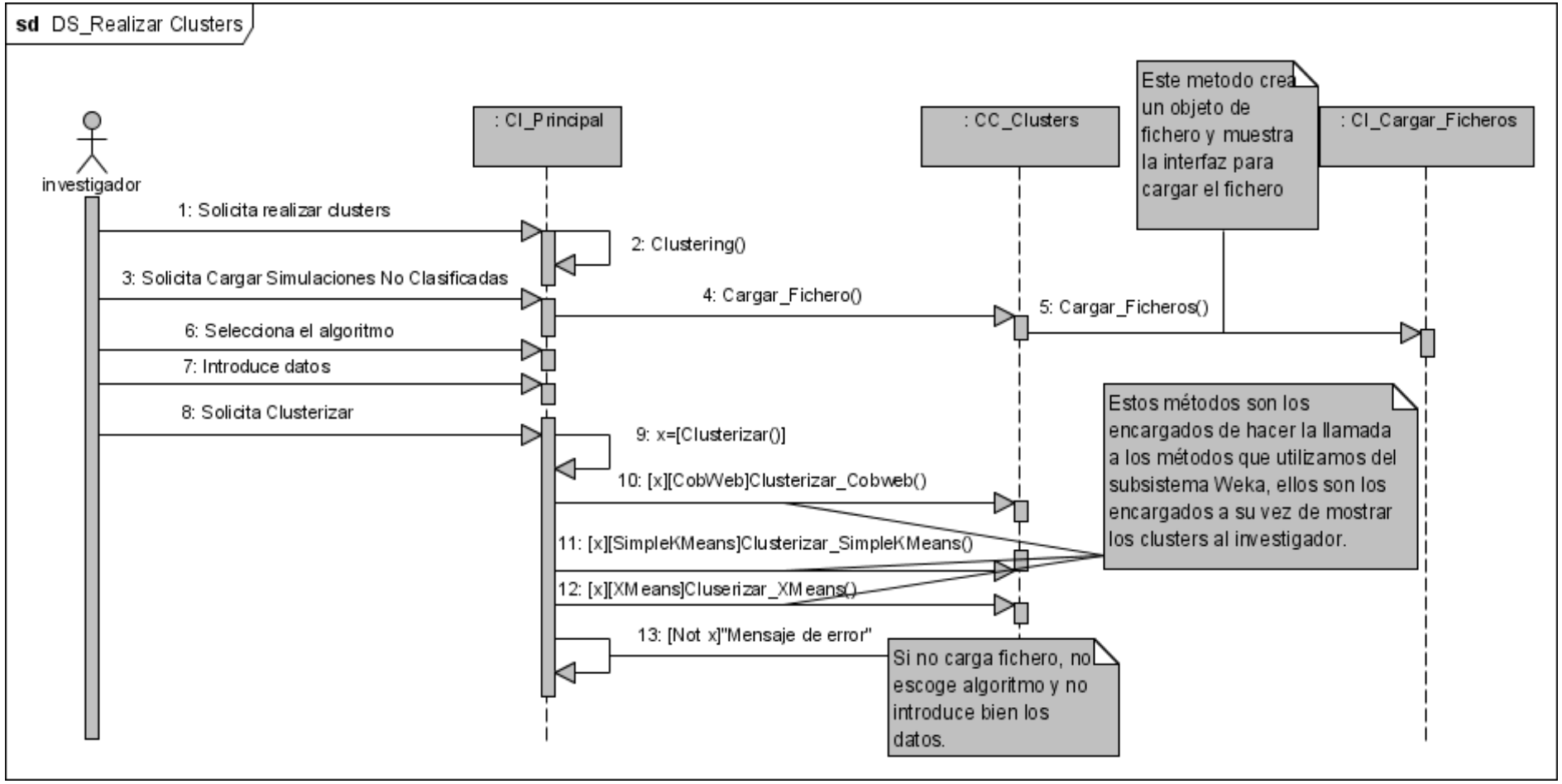


Figura 12. Diagrama de Secuencia de “Realizar Clusters”.

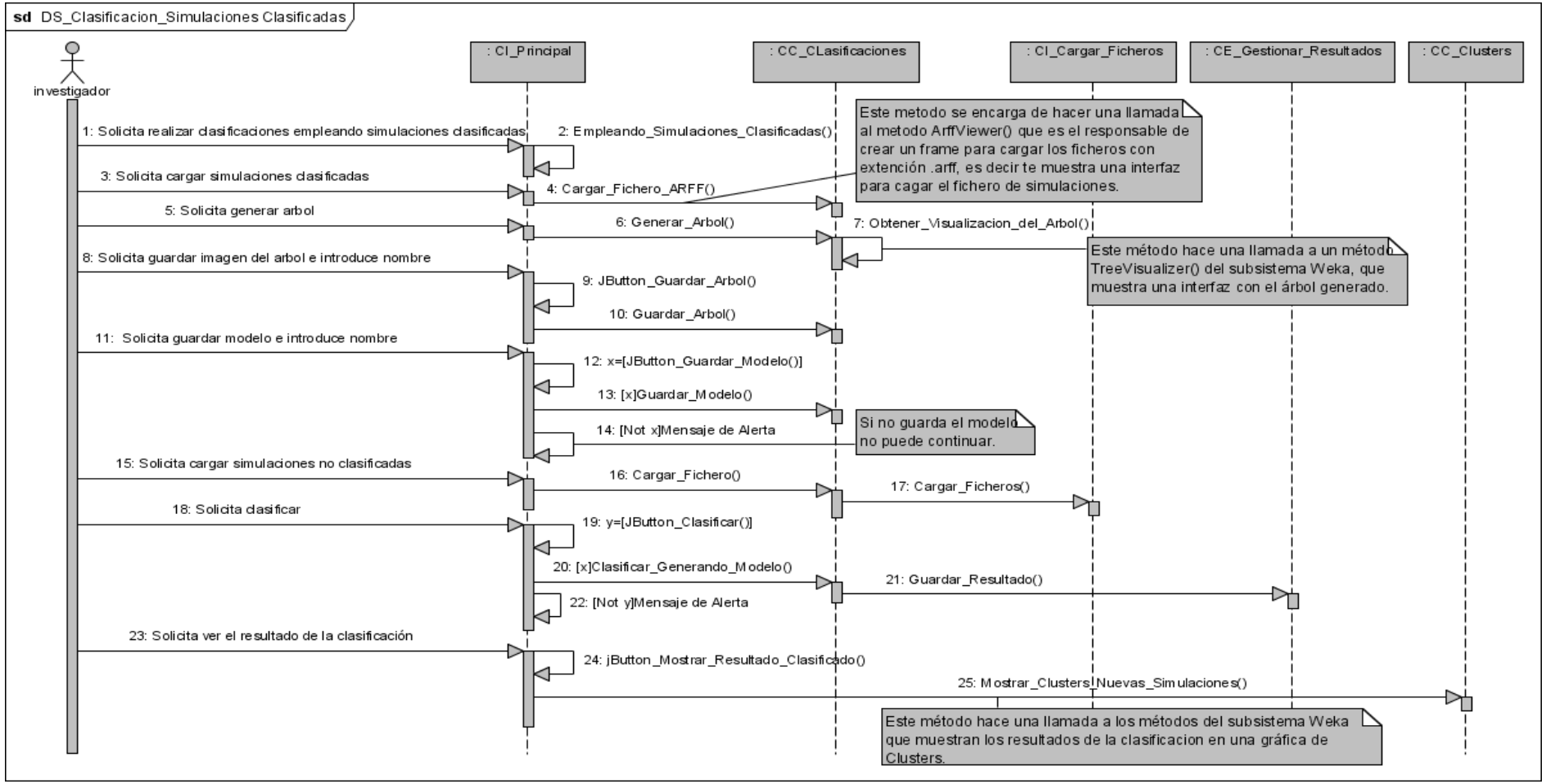


Figura 13. Diagrama de Secuencia de “Realizar Clasificaciones_Simulaciones_Clasificadas”.

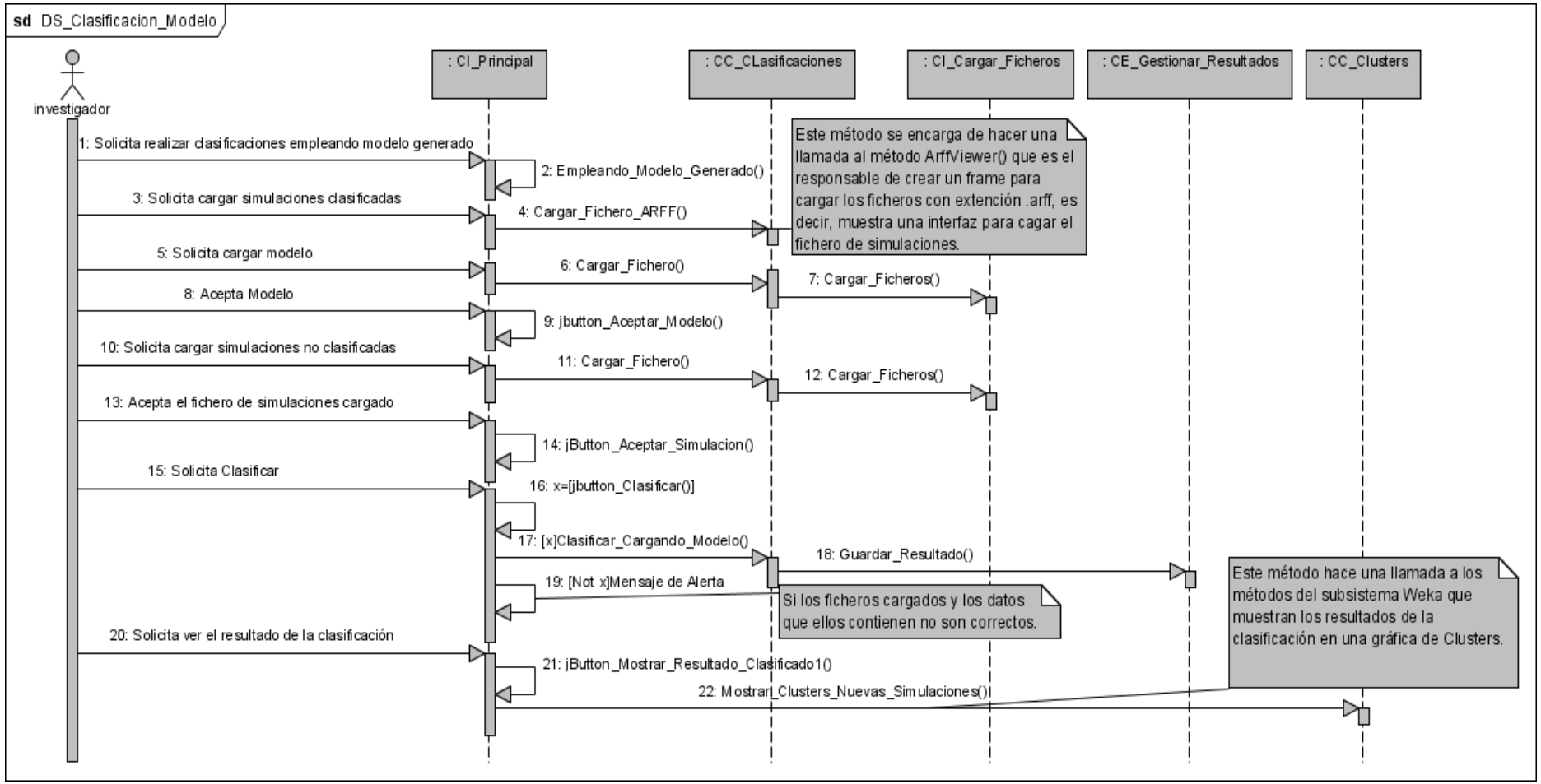


Figura 14. Diagrama de Secuencia de “Realizar Clasificaciones_Generando_Modelo”.

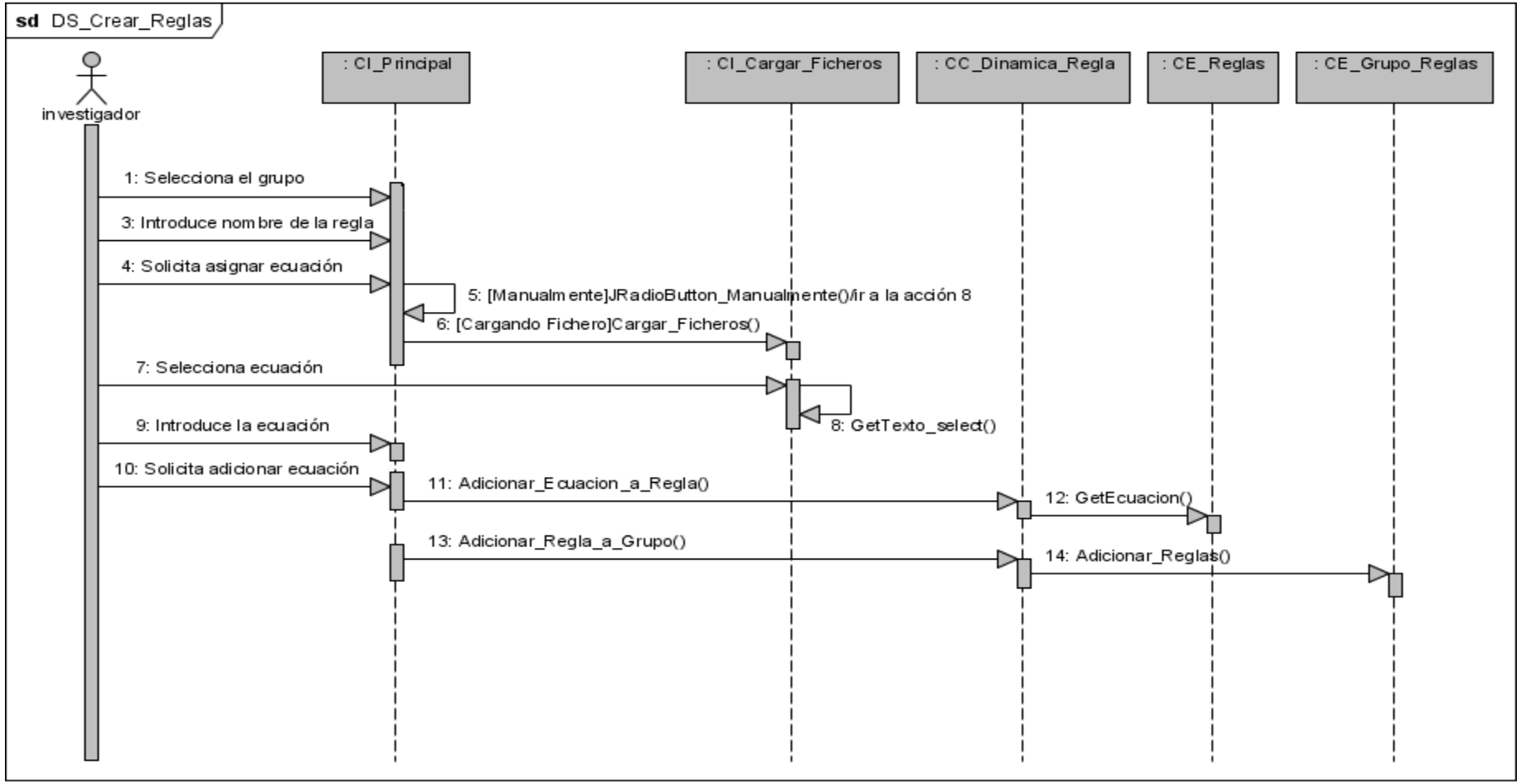


Figura 15. Diagrama de Secuencia de “Crear Reglas”.

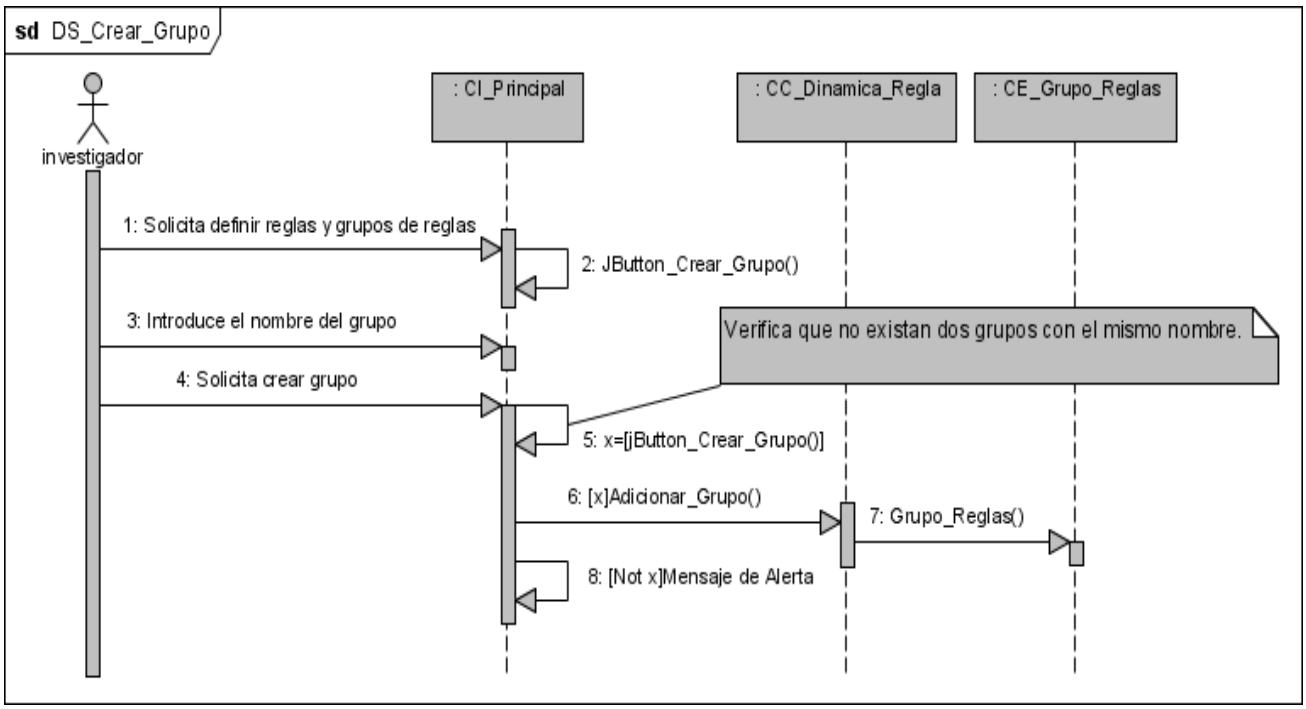


Figura 16. Diagrama de Secuencia de “Crear Grupos”.

CONCLUSIONES

En este capítulo se obtiene la estructura general del software, donde se aplicaron los patrones de arquitectura y diseño. Además de que se realizaron los diagramas de clases y de interacción para lograr que las clases, objetos, paquetes y otros elementos queden con la organización y detalle suficiente para que los programadores puedan desempeñar su rol correctamente.

CAPÍTULO 4: IMPLEMENTACIÓN DEL SISTEMA

Este capítulo contiene las principales características de la implementación del Sistema. Además de los principales componentes y sus relaciones, mostrados de forma sencilla a través del Diagrama de Componentes. También se muestran los resultados obtenidos con el código fuente y las interfaces más importantes de las principales clases empleadas en la implementación.

4.1 Diagrama de Componentes

Un Diagrama de Componentes contiene componentes, interfaces y relaciones entre ellos. Muestra las organizaciones y dependencias lógicas entre componentes de software. Normalmente los Diagramas de Componentes se utilizan para modelar código fuente, versiones ejecutables, bases de datos físicas, entre otros. Cada componente debe tener un nombre que lo distinga de los demás. La figura 17 muestra el Diagrama de Componentes correspondiente al Módulo de Análisis.

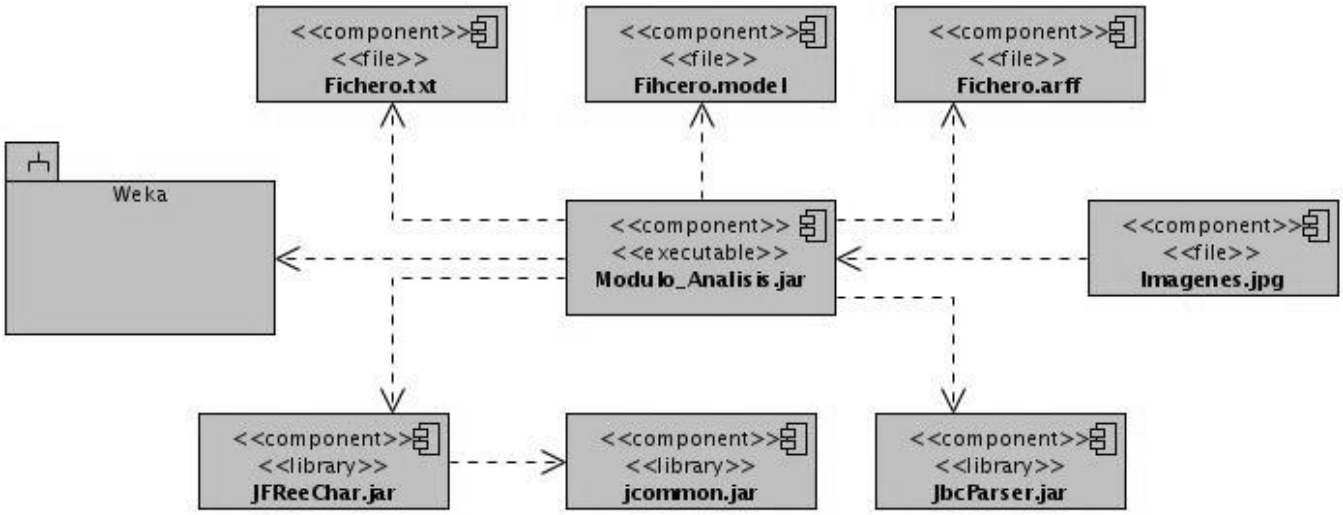


Figura 17. Diagrama de Componentes**.**

4.2. Resultados Obtenidos

En el presente trabajo se obtuvo una Aplicación de escritorio que permite a los investigadores realizar Análisis sobre los resultados obtenidos de las simulaciones. A continuación se explica lo que realiza cada uno de los métodos empleados para realizar el Análisis.

4.2.1 Dinámica de Poblaciones

Este método muestra como resultado una salida gráfica donde se representan las Series Temporales que se obtienen de las simulaciones que están siendo analizadas. De esta forma el investigador puede ver el comportamiento de cada población a lo largo del tiempo, para tomar decisiones. Aunque este es un análisis muy básico, es necesario para así poder comprobar si el modelo que se ha creado se ajusta a la realidad del problema en estudio. [Anexo 4]

4.2.2 Clustering

El análisis por clustering, o agrupamiento, utiliza el software Weka que tiene incluido una gran variedad de algoritmos, que fueron muy útiles en el desarrollo del trabajo, entre ellos utilizamos el X-Means, SimpleKMeans y el CobWeb, que se definieron desde la primera versión del Software. Este método muestra como resultado una salida gráfica donde se representan los Clusters, que agrupan los resultados de las simulaciones según el algoritmo seleccionado. [Anexo 5]

4.2.3 Clasificación

Este método utiliza el resultado de las simulaciones clasificadas anteriormente por los algoritmos de clusters para generar un clasificador, que no es más que un modelo que se genera utilizando el algoritmo J48(definida su utilización desde la primera versión de BioSyS), que proviene de los árboles de decisión que están incluidos en el Software Weka. Las clasificaciones se pueden realizar de dos formas:

1. A partir de ficheros que contienen los resultados de las simulaciones clasificadas.
2. A partir de modelos generados previamente por los resultados de simulaciones clasificadas.

Por cualquiera de estas dos vías se genera el clasificador que se utilizara para clasificar los resultados de las nuevas simulaciones no clasificadas. Finalmente se obtiene como resultado el fichero que contiene la clasificación de las simulaciones que no estaban clasificadas y se muestra una gráfica con el resultado de la clasificación. [Anexo 6]

4.2.4 Definir Reglas y Grupos de Reglas

Este método tiene como objetivo crear reglas y grupos de reglas permitiendo al investigador definir la ecuación lógica correspondiente a cada regla, esta ecuación lógica se puede definir de dos formas:

1. Editándola manualmente.
2. A través de ficheros que contienen ecuaciones.

Finalmente se obtiene ecuación que será asignada a la regla que has creado, estas reglas se emplearán en posteriores análisis por reglas.

4.3 Código Fuente

Tabla 6. Código fuente que permite graficar las Series Temporales.

```
public void Graficar(ArrayList<CE_Resultado> resultados, String[] a){
ArrayList<XYSeries> series = new ArrayList<XYSeries>();
for (int i = 0; i < a.length; i++){
series.add(new XYSeries(a[i]));
}
for (int i = 0; i < resultados.size(); i++){
series.get(resultados.get(i).GetIdpoblacion()).add(resultados.get(i).
GetTiempo(), resultados.get(i).GetValor());
}
XYSeriesCollection dataset = new XYSeriesCollection();
for (int i = 0; i < series.size(); i++){
dataset.addSeries(series.get(i));
}
JFreeChart chart = ChartFactory.createXYLineChart("Dinamica",
"Tiempo", "Poblaciones", ldataset, PlotOrientation.VERTICAL, true,
true, false); XYPlot plot2 = chart.getXYPlot();
XYLineAndShapeRenderer renderer2 = new XYLineAndShapeRenderer();
for (int i = 0; i < series.size(); i++){
renderer2.setSeriesLinesVisible(i, true);
renderer2.setSeriesShapesVisible(i, false);
}
renderer2.setToolTipGenerator(new StandardXYToolTipGenerator());
plot2.setRenderer(renderer2);
ChartPanel chartPanel = new ChartPanel(chart);
```

```
JFrame f = new JFrame(""); f.setSize(500, 500);
f.getContentPane().add(chartPanel);f.setVisible(true);
}
```

Tabla 7. Código fuente que se encarga de Clasificar.

```
public void Clasificar_Generando_Modelo(String nombre_modelo, String
direccion_sim_no_clasificadas) throws FileNotFoundException,
IOException{
Instances sim_no_clas = new Instances(new BufferedReader(new
FileReader(direccion_sim_no_clasificadas)));
Instances simulaciones_clasificadas = new Instances(new
BufferedReader(new FileReader(fichero_now)));
for (int i = 0; i < simulaciones_clasificadas.numAttributes(); i++)
{
if(simulaciones_clasificadas.attribute(i).isNominal())
sim_no_clas.insertAttributeAt(simulaciones_clasificadas.attribute(i)
,
sim_no_clas.numAttributes()); }
String result_a_guardar= direccion_proyecto +
"/data/Instancia_temporal_a_clasificar/temporal.arff" ;
CE_Gestionar_Resultados eliminar_temporal= new
CE_Gestionar_Resultados();
eliminar_temporal.Eliminar_Fichero_Direccion(result_a_guardar);
resultados_a_guardar=new CE_Gestionar_Resultados(result_a_guardar,
sim_no_clas.toString() );
resultados_a_guardar.Guardar_Resultado();
String comando=" -l " + direccion_proyecto + "/data/modelos/" +
nombre_modelo + ".model" + " -T " +
resultados_a_guardar.Direccion_fichero() + " -p 0" ;
try {
String [] commandArgs = Utils.splitOptions(comando);
 Evaluation eval= new Evaluation(temp);
```

```
resultado=eval.evaluateModel(new J48(), commandArgs);
Vector<String> temporal=getClusterInstancias(resultado);
for (int i = 0; i < sim_no_clas.numInstances(); i++) {
sim_no_clas.instance(i).setValue(
sim_no_clas.instance(i).numAttributes()-1,temporal.elementAt(i)); }
direccion_resultado_clasificacion = direccion_proyecto +
"/data/Resultados_Clasificacion/"+ nombre_modelo + ".arff";
CE_Gestionar_Resultados resultados_clacff= new
CE_Gestionar_Resultados(direccion_resultado_clasificacion
,sim_no_clas.toString() );
 resultados_clacff.Guardar_Resultado();}
catch (Exception ex) {ex.printStackTrace();}
}
```

Tabla 8. Código fuente que agrupa por el algoritmo SimpleKMeans.

```
public void Clusterizar_SimpleKmeans(int cantidad, String
direccion){
 try {
 SimpleKMeans algoritmo_SimpleKMeans=new SimpleKMeans();
algoritmo_SimpleKMeans.setNumClusters(cantidad);
Instances data = new Instances(new BufferedReader(new
FileReader(direccion)));
weka.filters.unsupervised.attribute.Remove filter = new
weka.filters.unsupervised.attribute.Remove();
filter.setAttributeIndices("" + (4));
filter.setInputFormat(data);
Instances dataClusterer = weka.filters.Filter.useFilter(data,
filter);
algoritmo_SimpleKMeans.buildClusterer(dataClusterer);
ClusterEvaluation eval = new ClusterEvaluation();
 eval.setClusterer(algoritmo_SimpleKMeans);
 eval.evaluateClusterer(dataClusterer);
plots= visualizar.setUpVisualizableInstances(data, eval);
```

```
VisualizePanel nuevo= new VisualizePanel();
nuevo.addPlot(plots);visualizar.visualizeClusterAssignments(nuevo);
} catch (Exception ex){
ex.printStackTrace();}
```

4.4 Interfaz de la Aplicación

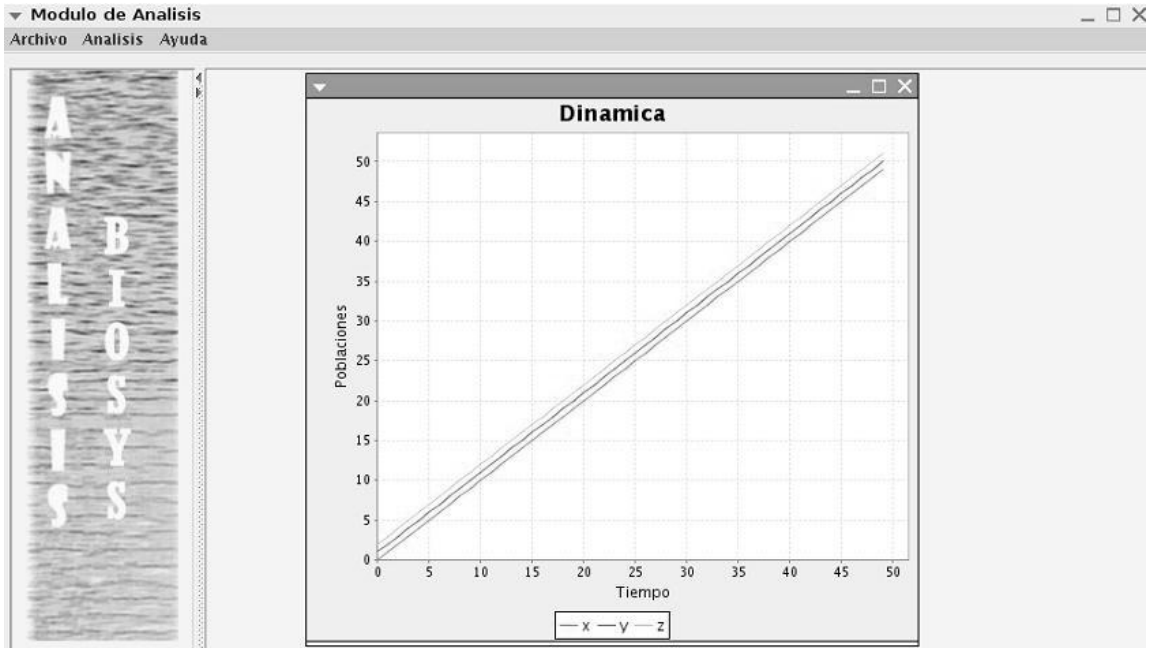


Figura 18. Interfaz para Mostrar las Dinámicas de Población.

Figura 19. Interfaz para Realizar Clusters.

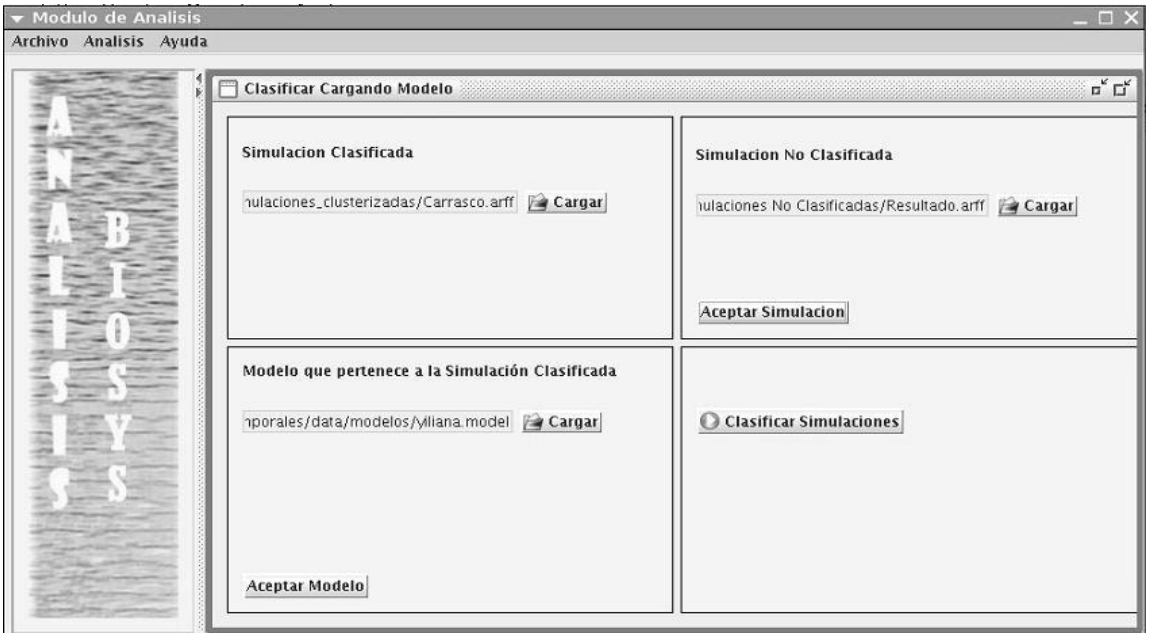


Figura 20. Interfaz para Realizar Clasificaciones Cargando Modelo.

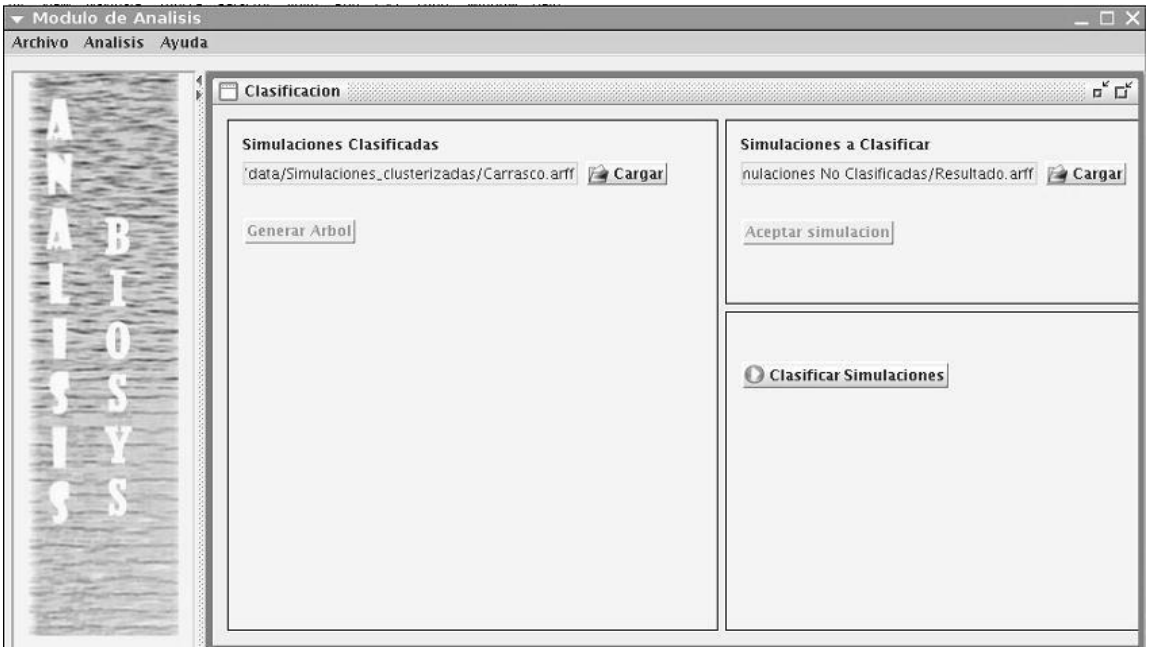


Figura 21. Interfaz para Realizar Clasificaciones Simulaciones Clasificadas.

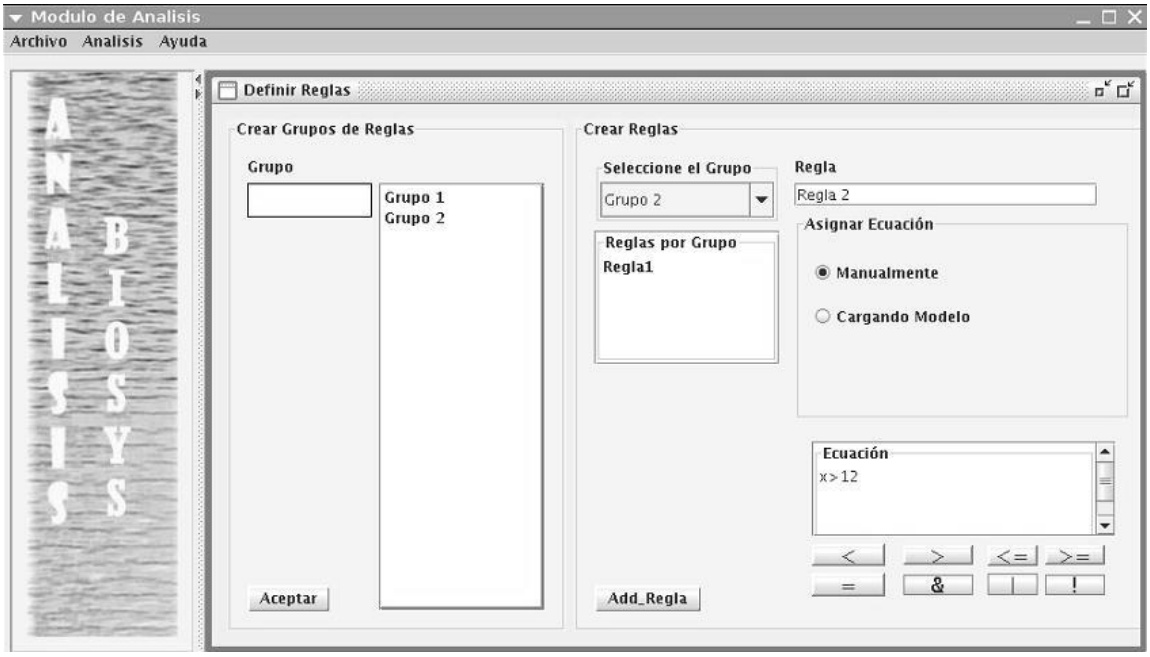


Figura 22. Interfaz para Definir Reglas**.**

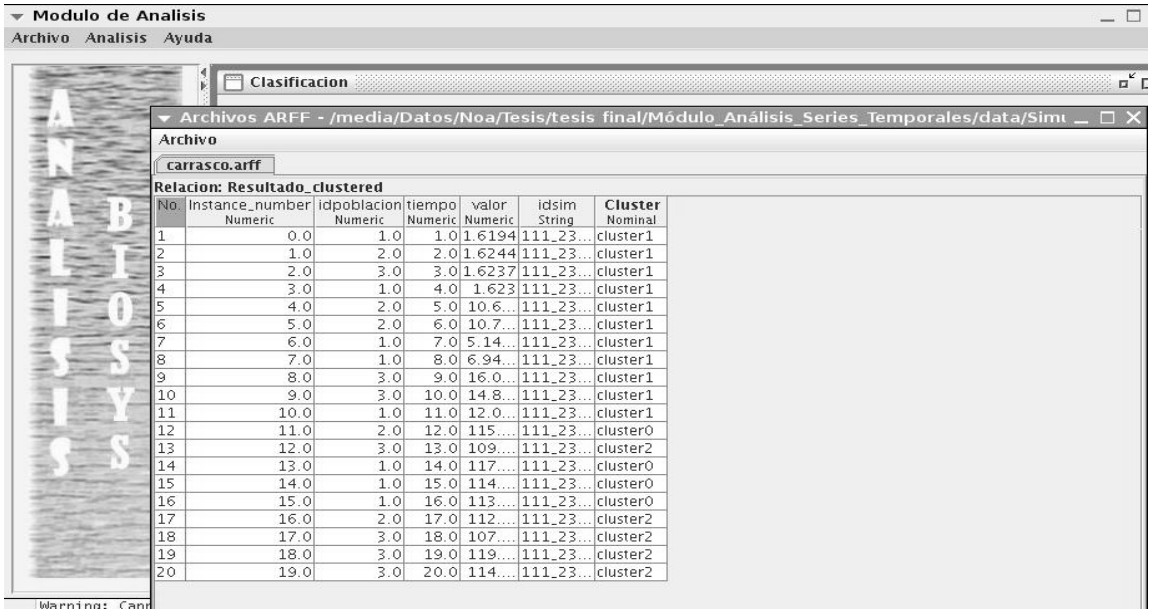


Figura 23. Interfaz para cargar y mostrar el fichero arff**.**

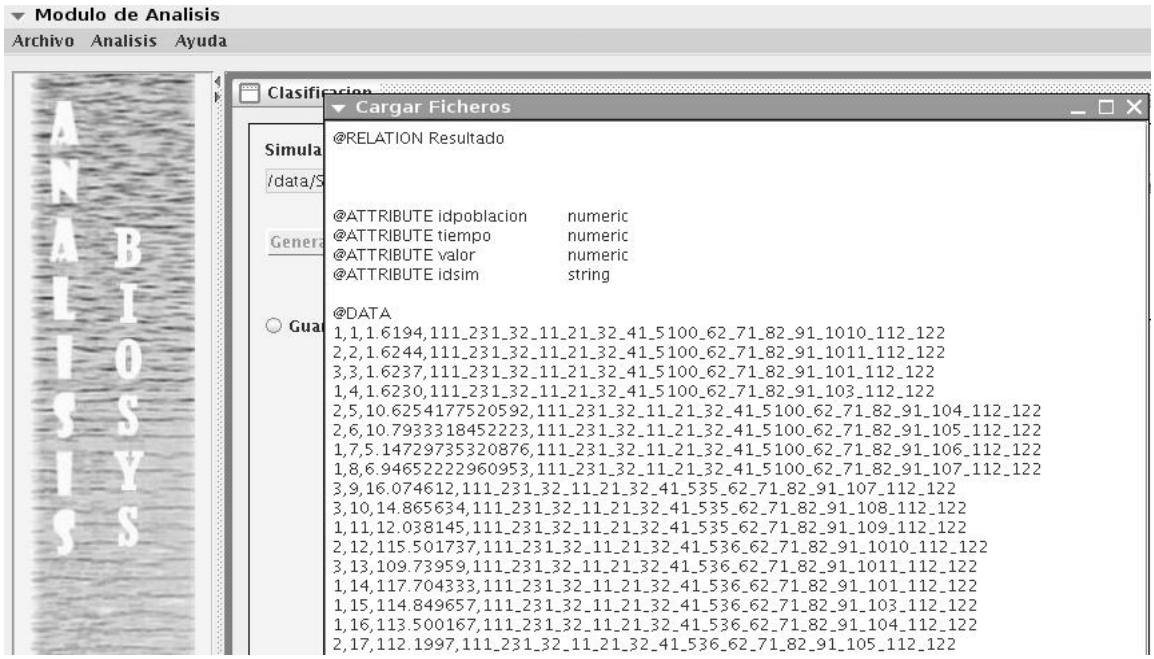


Figura 24. Interfaz para Cargar Fichero**.**

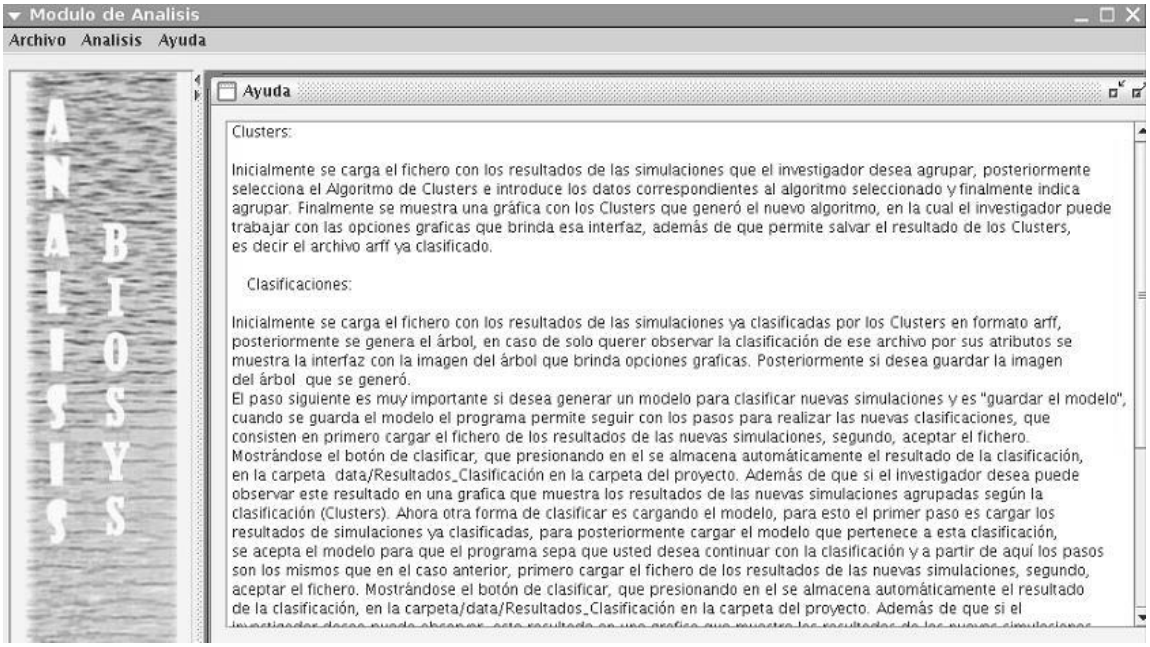


Figura 25. Interfaz que muestra la Ayuda.

4.5 Validación Funcional

Tabla 9. Caso de prueba Realizar Clasificaciones.

Caso de Uso	Realizar Clusters
Caso de Prueba	Obtener los Clusters
Entrada	Inicialmente se carga el fichero con los resultados de las simulaciones que el investigador desea agrupar, posteriormente selecciona el algoritmo de clusters e introduce los datos correspondientes al algoritmo seleccionado y finalmente indica agrupar. (Ver figura 26)
Resultado Esperado	Se obtiene una grafica que muestra los resultados de las simulaciones agrupadas. (Ver figura 27)
Resultado de la prueba	Al introducir los datos de Entrada y ejecutar el algoritmo de clusters el resultado de la prueba fue el esperado, mostrando la grafica con los resultados correctamente agrupados.

Condiciones	Debe cargar el fichero con los resultados de las simulaciones en formato arff, posteriormente seleccionar el algoritmo de clusters y finalmente entrar los datos correspondientes al algoritmo seleccionado, en caso de que estos datos de entradas no sean correctos se mostrarán mensajes de error brindando la posibilidad de que el investigador pueda entrarlos nuevamente.

Entrada:

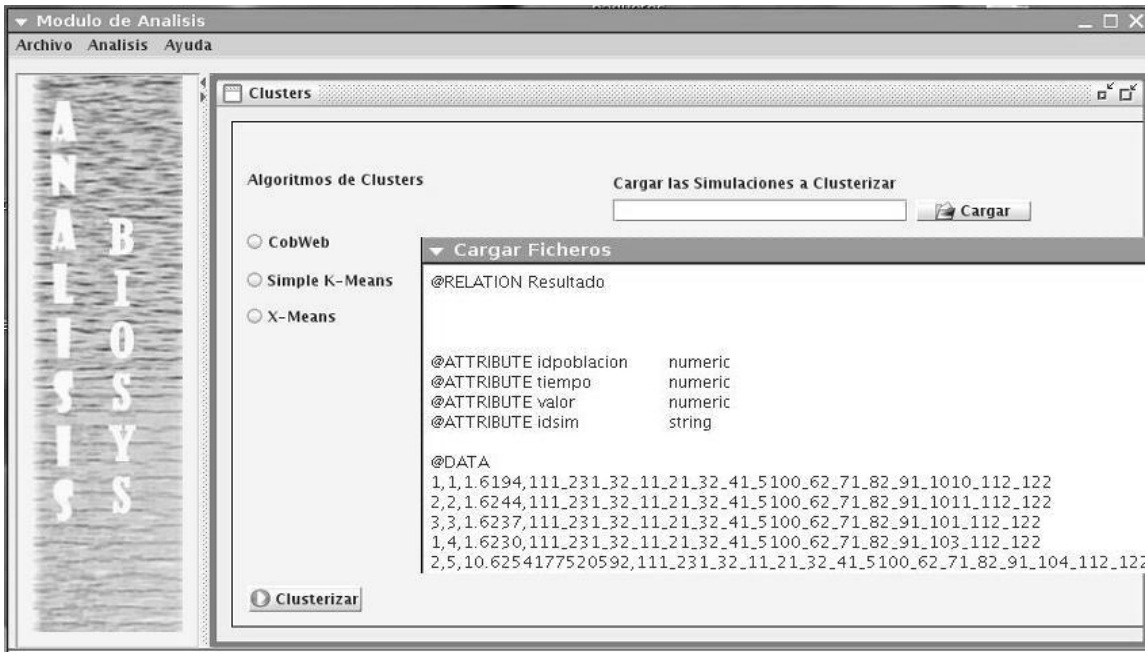


Figura 26. Datos de Entrada de “Realizar Clusters”.

Resultado:

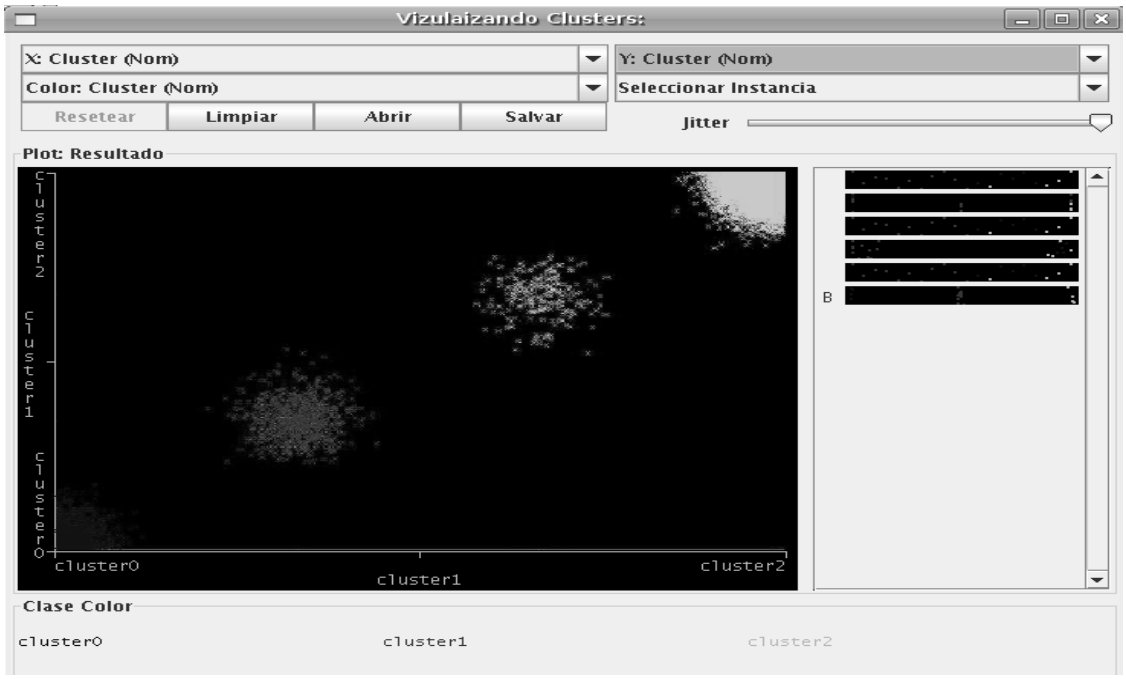


Figura 27. Resultado de “Realizar Clusters”

Tabla 10. Caso de prueba Realizar Clasificaciones.

Caso de Uso	Realizar Clasificaciones
Caso de Prueba	Obtener los nuevos resultados clasificados
Entrada	Inicialmente se carga el fichero con los resultados de las simulaciones ya clasificadas en formato arff, se guarda el modelo y finalmente se cargan los resultados de las simulaciones no clasificadas en formato arff. (Ver figura 28)
Resultado Esperado	Se obtiene una grafica que muestra los resultados de las nuevas simulaciones agrupadas según la clasificación (clusters). Además de que se guarda un fichero arff que contiene el resultado de las simulaciones no clasificadas y se le agrega la clasificación. (Ver figura 29)
Resultado de la prueba	Al introducir los datos de Entrada se obtiene el árbol de decisión esperado, también se guarda el modelo de forma correcta, finalmente se muestra la grafica de los nuevos resultados de las simulaciones ya

	clasificadas agrupados en la grafica que muestras los clusters y se guarda correctamente el fichero con las simulaciones ya clasificadas.
Condiciones	Debe cargar el fichero con los resultados de las simulaciones ya clasificadas en formato arff, posteriormente generar el modelo con el cual va a clasificar, guardarlo y cargar los resultados simulaciones no clasificadas en formato arff.

Entrada:

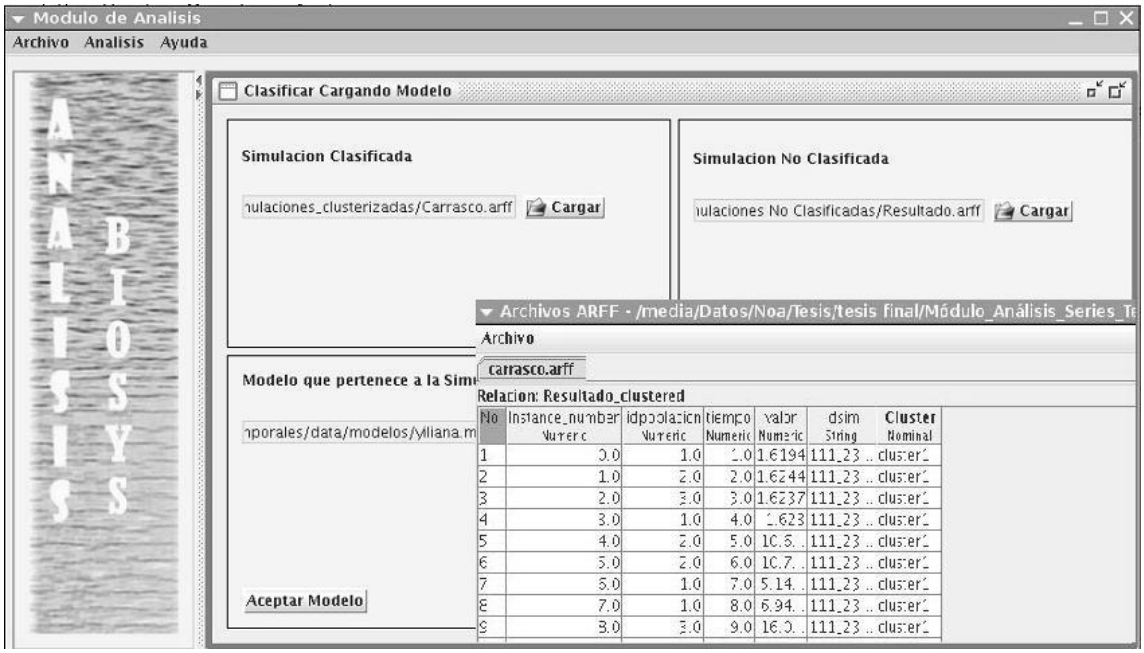


Figura 28. Datos de Entrada de “Realizar Clasificaciones”.

Resultado:

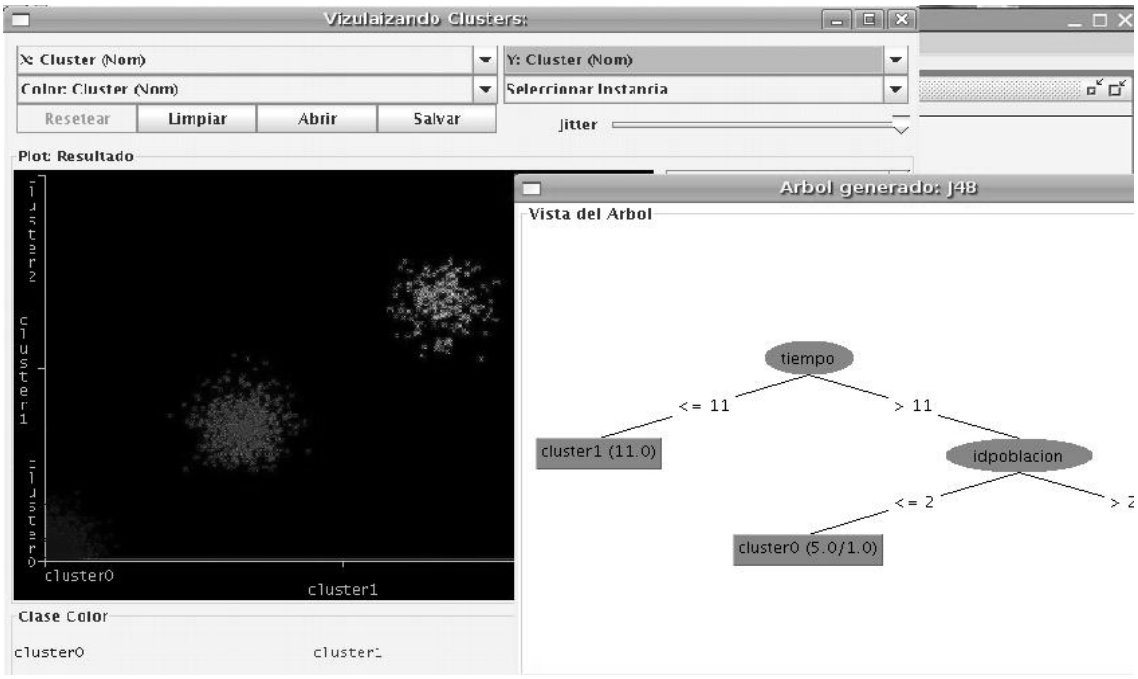


Figura 29. Resultado de “Realizar Clasificaciones”.

CONCLUSIONES

La implementación es una etapa fundamental para lograr materializar todo el proceso de creación de un software, este no es el final, pero es una parte muy importante para todo el equipo de trabajo involucrado. En este capítulo se presentaron los resultados obtenidos en el presente trabajo, y el código fuente de los métodos más importantes que reflejan las características que se definieron desde el inicio del proceso de creación de este software, para dar cumplimiento a los requisitos del cliente. Cumpliendo con los patrones y la arquitectura definidos.

CONCLUSIONES

El presente trabajo a lo largo de sus 4 capítulos mostró la importancia del Módulo de Análisis para el software BioSyS y para los investigadores, ya que le permite sacar las conclusiones de todo el proceso de Simulación que desarrolló el software.

1. Se identificaron los requerimientos funcionales y no funcionales del Módulo de AST.
2. Se diseñó el Módulo de AST, basándonos en la metodología propuesta y cumpliendo con los patrones de arquitectura y diseño especificados.
3. Se implementó el Módulo de AST, que permite mostrar los resultados de las Dinámicas de Poblaciones con el componente JfreeChar. Realizar Clusters a los resultados de las simulaciones utilizando tres algoritmos, CobWeb, SimpleKMeans y Xmeans, mostrando los clusters en una gráfica y guardando los resultados ya clasificados. También permite mostrar la Clasificación de los resultados de las simulaciones en forma de árbol de decisión, a través del algoritmo J48. Posibilita la clasificación de los nuevos resultados de simulaciones a través de un modelo generado previamente (Clasificador), por resultados de simulaciones ya clasificadas, luego, estos resultados se guardan y se muestran en una gráfica de clusters. Finalmente permite la creación de grupos de reglas y reglas con su correspondiente ecuación lógica para que sean usadas posteriormente en los Análisis por Reglas.

De forma general el trabajo de diploma ha posibilitado aumentar el nivel de conocimiento en áreas importantes de las ciencias biológicas y más específicamente en los distintos tipos de análisis que pueden ser aplicados a las simulaciones. Las experiencias adquiridas durante la realización del trabajo han sido excelentes, aportándonos conocimientos sobre temas muy novedosos, complicados e interesantes.

RECOMENDACIONES

1. Recomendamos la investigación de otros métodos importantes de análisis y el estudio de otros Algoritmos de Minería de Datos, para que los investigadores puedan tomar decisiones teniendo en cuenta los resultados que arrojen estas nuevas variantes y puedan hacer comparaciones interesantes.
2. La integración del Módulo de Análisis al Software BioSyS.
3. Implementar la realización de los análisis por reglas utilizando el componente JfreeChart. Para esta implementación ya se cuenta con algoritmos probados en la versión anterior (Bifurcaciones y Bisección).
4. Recomendamos de forma especial la continuación y mejora de este módulo, ya que los beneficios sociales que aporta a las investigaciones de hoy en día son innumerables. Es importante decir, que el aporte principal lo ha hecho ya, y es que, contribuye a las investigaciones en el Centro de Inmunología Molecular(CIM) en el estudio y la obtención de posibles medicamentos y vacunas para un sin número de enfermedades y padecimientos.

REFERENCIAS

1. **B.Tidor., B.Tadmor and.** *Interdisciplinary research ande education at the biologu engineering computer science interface: a perspective(reprinted article).*Biosilico, Vol 10(No.17). September 2005.
2. *Unidad I: Conceptos Básicos de Simulación. C.A*
3. **Lic. Francisco García Mora, Dr. C. F. Jorge Sierra Acosta,María Virginia Guzmán.** *10ma Convención Internacional de las Industrias Metalúrgica, Metalmecánica y del Reciclaje. XI Taller de Gestión Tecnológica en la Industria.* Palacio de Convenciones de La Habana, Cuba : s.n., 08 al 12 de octubre de 2007.
4. **Catalán, Cecilia Esparza.** *Series Temporales.* http://estadistica.ieg.csic.es/tutoriales/PDF/SeriesTemporales.pdf
5. **M.Molinero, Luis.** *Análisis de Series Temporales.* Enero 2004. http://www.sehlelha.org/tseries.htm
6. **Meliá, J.L.** *Análisis de Datos con BMDP.* 1997.
7. **Lemus, Noel Moerno.** *Tesis de Maestria. Simulador de Sistemas Biológicos* (BioSyS). Cuba.La Habana : s.n., 2007.
8. http://diccionarios.elmundo.es/diccionarios/cgi/lee_diccionario.html?busca=particion&submit=+Buscar+&diccionario=1
9. http://10.34.20.5:5800/OpenUP/
10. **Grupo de Autores.** *Arquitectura de software. Capítulo 4.* 4 de Abril del 2004.
11. *Patrones de diseño.Diseño de Software Orientado a Objetos*. *Por Joaquín Graci.27 de Mayo de 2005.* www.ingenierosoftware.com
12. OpenUp. [En línea] http://10.34.20.5:5800/OpenUP/
13. **Morate, Diego García.** *Manual de Weka.*
14. http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3317/default.aspx
15. **Larman, C**; *"UML y patrones" Tomo I* Capítulos 18, Páginas 185-215
16. **Robaina, Irilis Ledón.** *Tesis de Diploma de Arquitectura de BioSyS.* Cuba. La Habana : s.n., 2008.
17. http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=software

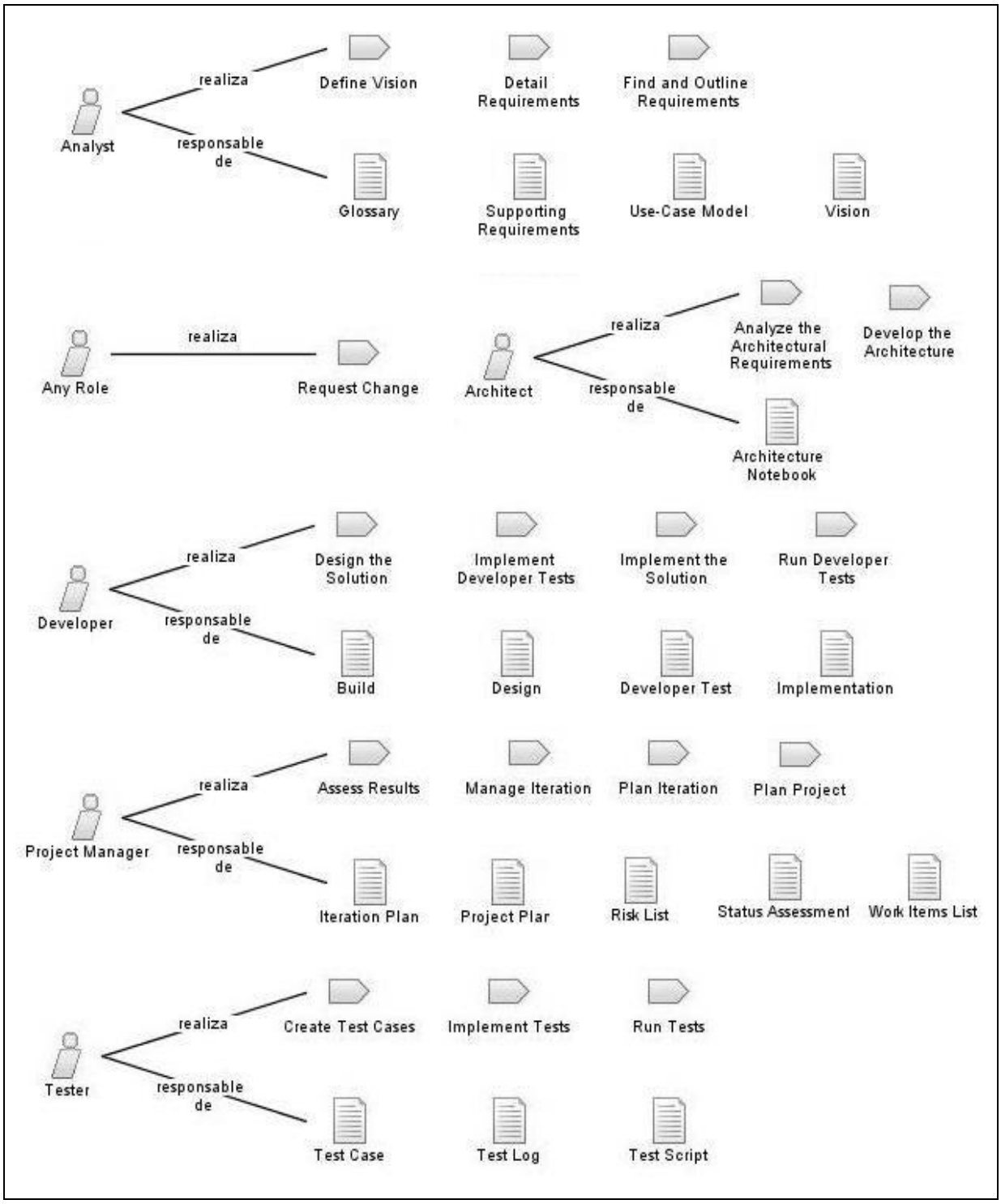
BIBLIOGRAFIA

1. **Robaina, Irilis Ledón.** *Tesis de Diploma de Arquitectura de BioSyS.* Cuba. La Habana : s.n., 2008.
2. **Maraval, Agustín.** *BRIEf DESCRIPTION OF THE PROGRAMS*. May 2005.
3. **Lemus, Noel Moerno.** *Tesis de Maestria. Simulador de Sistemas Biológicos* (BioSyS). Cuba.La Habana : s.n., 2007.
4. **Delgado, Yunet Gonzales Mulet y Yanet Alonso.** *Trabajo de Diploma. Software para la Simulación de Sistemas Biológicos*. Cuba.La Habana : s.n., 2007.
5. **M.Molinero, Luis.** *Análisis de Series Temporales.* Enero 2004. www.seh-lelha.org/tseries.htm
6. **Fernández, Jesús Sánchez.** *Introducción a la Estadística Empresarial. Capítulo 4 Series Temporales.*
7. **Catalán, Cecilia Esparza.** *Series Temporales.*
8. **S.Pressman, Roger.** *Ingenieria de Software. Un enfoque práctico Parte I. .*
9. **Autores, Grupo de.** *Arquitectura de software.Capítulo 4. .* 4 de Abril del 2004.
10. **Autores, San Sebastian. Colectivo de.** *Aprenda Java como si estuviera en primero .* Enero 2000.
11. **Balduino, Ricardo.** *Basic Unified Process: A Process for Small and Agile Projects. .*
12. OpenUp. [En línea] http://10.34.20.5:5800/OpenUP/
13. *Análisis de Series Temporales. Tema 10.*
14. **Morate, Diego Garcia.** *Manual de Weka.*
15. **Norambuena, Alfredo Gómez.** *Manual WEKA Explorer.*
16. **Carrasco, Roberto Portugal & Miguel A.** *Ensamble de Algoritmos Bayesianos con Arboles de Decisión. Una Alternativa de Clasificación.*
17. **Ramírez, César Ferri.** *Práctica de Minería de Datos. Introducción al Weka. Curso de Doctorado Extracción Automática de Conocimiento en Bases de Datos e Ingeniería del Software.* Marzo 2006.
18. **J.L. Cubero, F. Berzal, F. Herrera. Master oficial de la Universidad de Granada.** *Fundamentos de Minería de Datos "Soft COMPUTING Y SISTEMAS INTELIGENTES".*
19. *Programa de Doctorado. Tecnologías Industriales. Aplicaciones de la Inteligencia Artificial en Robótica. Práctica 1: Entorno WEKA de aprendizaje automático y data mining.*
20. **Meliá, J. L.** *Análisis de datos con BMDP.* 1997.

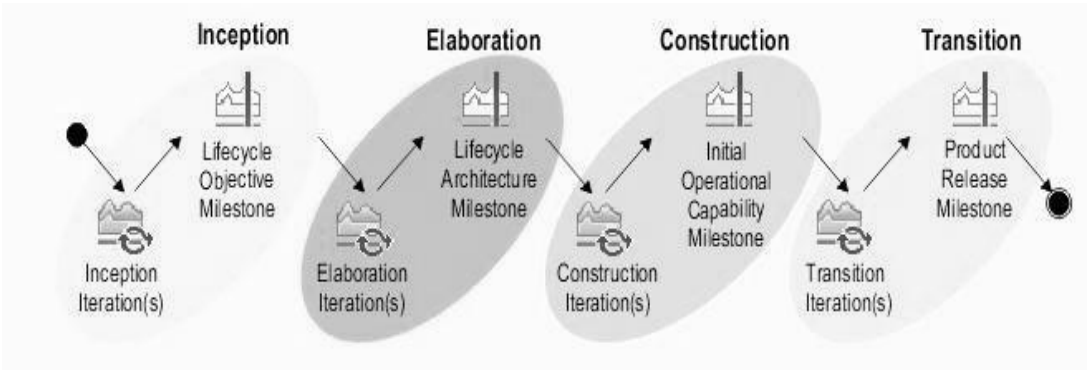
21. **María N. Moreno García∗, Luis A. Miguel Quintales, Francisco J. García Peñalvo y M. Jo.** *Aplicación de Técnicas de Minería de Datos en la Construcción y Validación de Modelos Predictivos y Asociativos a partir de Especificaciones de Requisitos de Software.*
22. **García Jiménez, M.V.** *Justificación del empleo del modelo ARIMA para Series Temporales en las Ciencias Sociales y de la Salud.*
23. **Pedro Larrañaga, Iñaki Inza, Abdelmalik Moujahid.** *Tema 14. Clustering.(Departamento de Ciencias de la Computación e Inteligencia Artificial).*
24. **Mora, Francisco García.** *10ma Convención Internacional de las Industrias Metalúrgica, Metalmecánica y del Reciclaje.*
25. **Lic. Francisco García Mora, Dr. C. F. Jorge Sierra Acosta,María Virginia Guzmán.** *10ma Convención Internacional de las Industrias Metalúrgica, Metalmecánica y del Reciclaje. XI Taller de Gestión Tecnológica en la Industria.* Palacio de Convenciones de La Habana, Cuba : s.n., 08 al 12 de octubre de 2007.
26. **B.Tidor., B.Tadmor and.** *Interdisciplinary research ande education at the biologu engineering computer science interface: a perspective(reprinted article).Biosilico, Vol 10(No.17).* September 2005.
27. **Graci, Joaquin.** *Patrones de diseño.Diseño de Software Orientado a Objetos.* 27 de Mayo de 2005. www.ingenierosoftware.com
28. microsoft.com. [En línea] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3317/default.aspx
29. diccionarios.elmundo.es. [En línea] http://diccionarios.elmundo.es/diccionarios/cgi/lee_diccionario.html?busca=particion&submit=+Buscar+&diccionario=1
30. OpenUp. [En línea] http://10.34.20.5:5800/OpenUP/
31. cs.waikato.ac.nz. [En línea] [Citado el: Miercoles 21 de Noviembre de 2007.] http://www.cs.waikato.ac.nz/ml/weka/
32. java.ciberaula.com. *Patrones de Diseño en aplicaciones Web con Java J2EE (ARTICULO).* [En línea] http://www.java.ciberaula.com/articulo/diseno_patrones_j2ee
33. quedque.es. [En línea] [Cited: Abril Sabado 19, 2008.] http://quedque.es/search.php
34. sparxsystems.com.ar. [En línea] [Cittado: Abril Lunes 17, 2008.] http://www.sparxsystems.com.ar/resources/tutorial/physical_models.html
35. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=jfreechart

ANEXOS

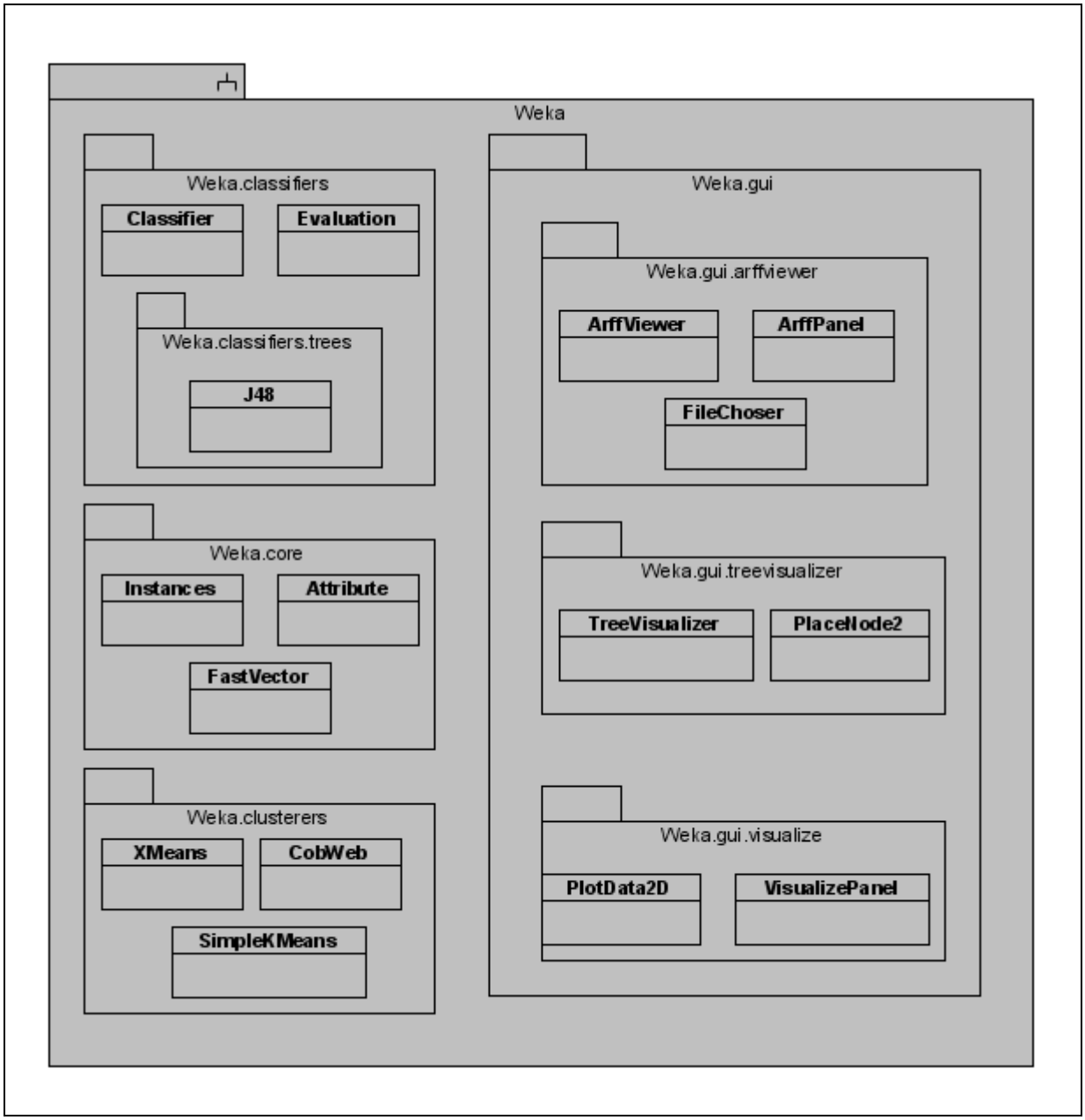
Anexo 1: Roles, Tareas y Artefactos de la Metodología OpenUP



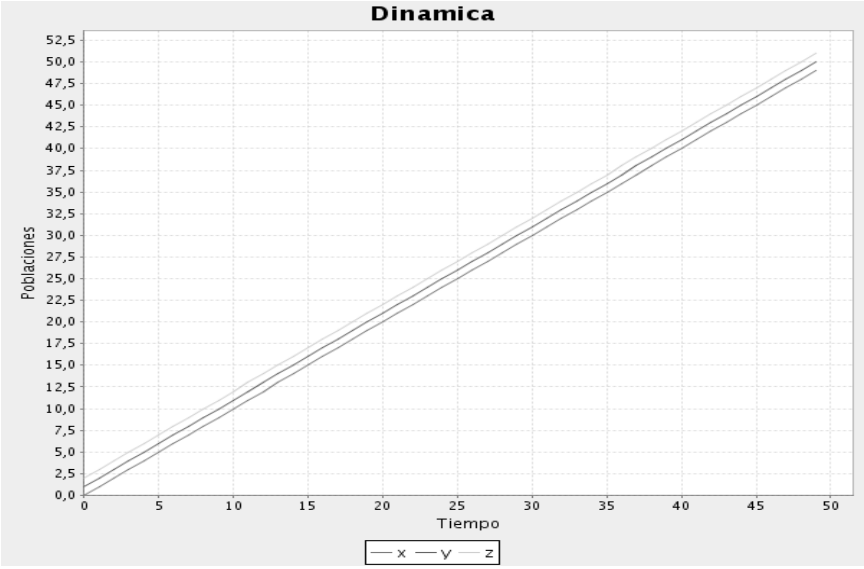
Anexo 2: Flujos de Trabajo de OpenUP



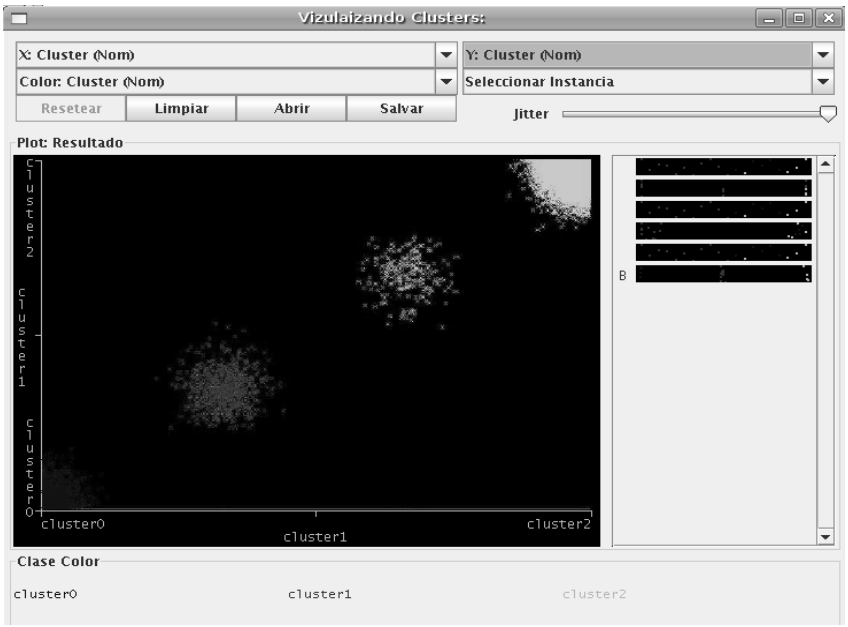
Anexo 3: Algunas clases del Weka



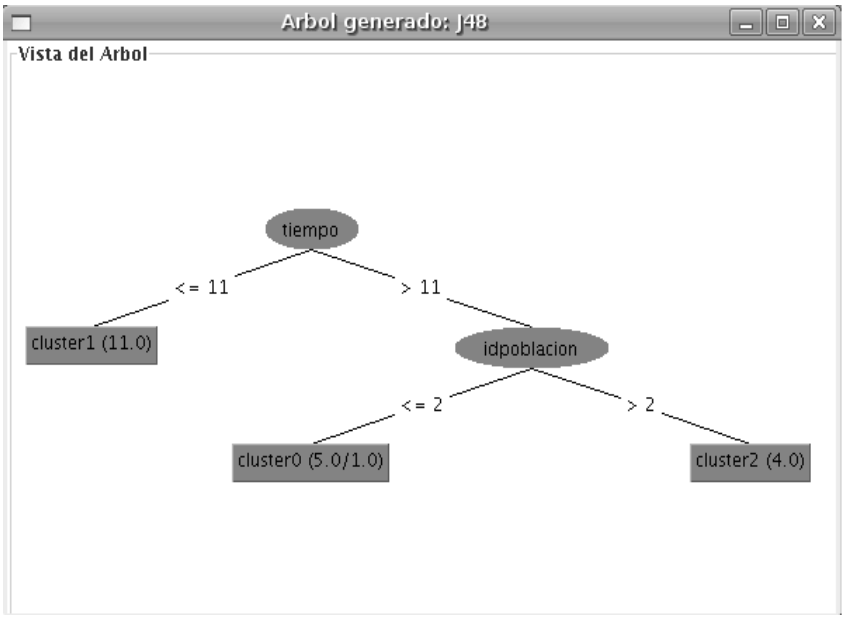
Anexo 4: Dinámica de Poblaciones



Anexo 5: Clustering



Anexo 6: Clasificaciones



GLOSARIO DE TERMINOS

Algoritmo: Es una lista bien definida, ordenada y finita de operaciones que permite hallar la solución a un problema.

Análisis: Estudio, mediante técnicas informáticas, de los límites, características y posibles soluciones de un problema al que se aplica un tratamiento por ordenador.

ARIMA: Modelo Auto-regresivos Integrados con Media Móvil. Modelo que permite pronosticar el comportamiento de una serie de tiempo.

AST: Análisis de Series Temporales. Comprende métodos que ayudan a interpretar los datos, extrayendo información representativa para predecir su comportamiento futuro.

BioSyS: Biological System Simulator (Software para la Simulación de Sistemas Biológicos).

Centroide: Centro de masa de un objeto con densidad uniforme. El centroide de un cluster se define como el punto equidistante de los objetos pertenecientes a dicho cluster, es como decir el punto de equilibrio.

Clasificación: Es la acción o el efecto de ordenar o disponer por clases.

Cluster: Grupo.

Clustering: Es la clasificación de los objetos en diferentes grupos.

CU: Caso de Uso

GNU/GPL: Es la Licencia Pública General de GNU orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

MD: Minería de Datos.

Partición: División o reparto que se hace entre varios.

SED: Sistema de Ecuaciones Diferenciales.

Serie Temporales: Sucesión ordenada, a través del tiempo, de un conjunto de variables aleatorias.

Simulación: Presentación de algo como real. Es un intento de modelar situaciones de la vida real por medio de un programa de computadora.

Software: Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora [17]

Stakeholders: Representa un rol en la metodología de desarrollo, el mismo podría ser desempeñado por cualquiera que esté (o potencialmente estará) materialmente afectado por el resultado del proyecto. Además representa grupos de interés cuyas necesidades deben ser satisfechas por el proyecto.